Decentralized Anonymous Micropayments*

Alessandro Chiesa[1], Matthew Green[2], Jingcheng Liu[1], Peihan Miao[1], Ian Miers[2] and Pratyush Mishra[1]

[1] UC Berkeley
{alexch,peihan,liuexp,pratyush}@berkeley.edu
[2] Johns Hopkins University
{mgreen,imiers}@cs.jhu.edu

Abstract. Micropayments (payments worth a few pennies) have numerous potential applications. A challenge in achieving them is that payment networks charge fees that are high compared to "micro" sums of money.

Wheeler (1996) and Rivest (1997) proposed probabilistic payments as a technique to achieve micropayments: a merchant receives a macro-value payment with a given probability so that, in expectation, he receives a micro-value payment. Despite much research and trial deployment, micropayment schemes have not seen adoption, partly because a trusted party is required to process payments and resolve disputes.

The widespread adoption of decentralized currencies such as Bitcoin (2009) suggests that decentralized micropayment schemes are easier to deploy. Pass and Shelat (2015) proposed several micropayment schemes for Bitcoin, but their schemes provide no more privacy guarantees than Bitcoin itself, whose transactions are recorded in plaintext in a public ledger.

We formulate and construct *decentralized anonymous micropayment* (DAM) schemes, which enable parties with access to a ledger to conduct offline probabilistic payments with one another, directly and privately. Our techniques extend those of Zerocash (2014) with a new privacy-preserving probabilistic payment protocol. One of the key ingredients of our construction is *fractional message transfer* (FMT), a primitive that enables probabilistic message transmission between two parties, and for which we give an efficient instantiation.

Double spending in our setting cannot be prevented. Our second contribution is an economic analysis that bounds the additional utility gain of any cheating strategy, and applies to virtually any probabilistic payment scheme with offline validation. In our construction, this bound allows us to deter double spending by way of advance deposits that are revoked when cheating is detected.

1 Introduction

We formulate and construct *decentralized anonymous micropayments*, by way of probabilistic payments.

Micropayments. A *micropayment* is a payment of a small amount, e.g., a fraction of a penny [Whe96, Riv97]. Micropayments have many potential applications, including advertisement-free content delivery, spam protection, rewarding nodes of P2P networks,

* This work was supported in part by the Center for Long-Term Cybersecurity at UC Berkeley.

and others. Achieving micropayments involves at least two main challenges. First, payment processing fees dwarf "micro" payment values. Second, micropayment applications often require *fast merchant responses*, which, in many settings, are achieved via *offline payments*, which are vulnerable to double spending.

Probabilistic payments. A technique to reduce processing fees is to amortize them over multiple payments by way of *probabilistic payments* [Whe96, Riv97].[3] These are protocols that enable a customer to pay V units of currency to a merchant with probability p: with probability $1 - p$ the merchant receives a *nullpayment* that is not processed, and with probability p the merchant receives a *macropayment* that is processed. In expectation, the merchant receives pV units per micropayment, but the overhead and processing fees of these "lottery tickets" is p times smaller as only the infrequently generated macropayments are actually handled by the payment network. Constructing probabilistic payments is an area of ongoing interest in cryptography.

Centralized vs. decentralized systems. Despite extensive research and trial deployments [Whe96, Riv97, LO98, MR02, Riv04, Mic14], micropayment schemes have not seen widespread usage. This is perhaps due to them being *centralized systems*: a trusted third party is tasked with processing payments and punishing cheaters. Appointing such a party raises deployment costs, requires establishing complex business relationships between all involved (the trusted party, merchants, and customers), and makes participation conditional on certain requirements being met [vOR+03].

Recent work in digital currencies has focused on *decentralized systems*, as the cost of entry and deployment appears to be lower. The most notable such currency is Bitcoin [Nak09], a widely adopted peer-to-peer payment system. Unlike traditional banking and e-cash schemes [Cha82, CHL05, ST99] where transactions are processed by a trusted party, Bitcoin utilizes a distributed public ledger known as the *blockchain* to store all transactions; these transactions are verified by network nodes in a peer-to-peer fashion.

Decentralized systems are thus potentially attractive for micropayments, because the overhead involving trusted parties is no longer a factor. However, Bitcoin processing fees are still relatively high (as of May 2016 the fee for a 1kB-transaction is $\approx \$0.20$), with present fees believed to be well below the cost of performing a transaction on the Bitcoin network [MB15]. Thus, fee amortization is still necessary. Caldwell [Cal12] first sketched probabilistic payments for Bitcoin. Recently, Pass and Shelat [PS15, PS16] also proposed three probabilistic payment schemes for Bitcoin, where, informally, the customer first puts V bitcoins in escrow, and then the customer and merchant engage in a coin-flipping protocol that allows the merchant to retrieve the escrow with probability p. Their three schemes differ in how payments are processed and how disputes are resolved.

Our privacy goal and limitations of prior work. We study the question of how to construct *decentralized anonymous micropayments* via the technique of (offline) probabilistic payments. The aforementioned prior work [PS15, PS16] provides *no more privacy than the underlying Bitcoin protocol*. And Bitcoin itself provides little to no privacy because every transaction is publicly broadcast and contains a payment's origin, destination, and amount; a user's payment history is thus readily available to any passive

[3] Another technique is micropayment channels, which we discuss in Section 1.2.

observer who can can link pseudonyms together or to real world identities.[4] This lack of privacy is particularly dangerous for micropayment applications because they typically involve high-volume pattern-rich payments (e.g., per-click payments while surfing the web), and sometimes necessitate user anonymity (e.g., bandwidth payments for Tor relays [BP15]).

Privacy is not merely an issue of individual users: if each coin's history is public, a customer may not be able to spend a coin at its 'declared' value due to its past. For example, a merchant may not accept coins whose past owners include certain political organizations. Privacy thus ensures a fundamental property of the currency: *fungibility*, which means that any two sets of coins with the same 'declared' total value are interchangeable, regardless of their provenance.

Prior work on privacy-preserving analogues of Bitcoin [MGGR13, DFKP13, BCG+14] does not achieve probabilistic payments, and merely "plugging" these schemes into [PS15, PS16]'s approach results in subtle problems. Consider the following natural modification to Pass and Shelat's coin-flipping protocol: instead of a Bitcoin transaction, the sender probabilistically transmits to the merchant a Zerocash transaction [BCG+14]. Despite the strong anonymity guarantees provided by Zerocash, merchants *still* learn information about their customers' spending habits, because each Zerocash transaction includes a unique serial number corresponding to the spent "coin". Since the customer sends to the merchant information about the escrow, this serial number is revealed in each micropayment. Since the same escrow is used across multiple probabilistic payments (to amortize fees), privacy of the customer is compromised because the merchant learns (1) which (macro or null) payments to him were made with the same escrow; and (2) which macropayments to other merchants were made with an escrow used for payments to him. This breach of privacy worsens if merchants share information with one another. In sum, while the above natural approach achieves "macropayment unlinkability", *micropayments are still linkable*, and thus customers have little privacy.

Double spending in offline probabilistic payments. Micropayment applications often require fast responses. In many settings, these in turn require *offline* validation: a merchant responds to a payment after only a local "offline" check, because he cannot wait for the payment network to validate the payment (this validation instead completes after the merchant's response). For example, validation takes a few minutes in Bitcoin, while responding to unconfirmed `zero-conf` transactions takes only a few seconds. We thus focus on *offline probabilistic payments*.

However, such payments are *vulnerable to double spending*, as we now explain. First, double spending *cannot be prevented* for offline payments, because, to prevent it, a merchant would have to refrain from responding to any payment before all payments up to, and including, this payment have been validated. One fallback is to detect and punish all double-spending customers. However, for offline *probabilistic* payments, not all double spending can even be detected.

Indeed, there are two types of double spending when using the same lottery ticket in two probabilistic payments: (1) both payments result in macropayments; or (2) the first

[4] This is not merely a theoretical concern: extracting information from Bitcoin transactions is the subject of applied research [RH11, BBSU12, RS13, MPJ+13] and commercial ventures [Ell13, Blo14, Cha15].

payment results in a macropayment (thereby 'consuming' the ticket) while the second payment results in a nullpayment. While detecting the first type is easy, detecting the second type requires the payment network to 'know' the temporal order of all payments, because whether the nullpayment or the macropayment occurred first determines whether the two payments correspond to honest behavior (nullpayment first) or not (macropayment first). But knowing the global order of all payments (with high precision) is a strong synchronization property that is unrealistic in many decentralized settings, including that of Bitcoin, because information does not instantly reach everyone in the network.

Given that not all double spending can be detected, the "detect-and-punish" approach is effective only if the disadvantages of being punished (upon detection) outweigh the advantages of double spending. This may be plausible in the centralized setting, where customers have registered with a trusted party that can permanently ban and legally prosecute them. In the decentralized setting, however, banning has few consequences, if any: anyone can abandon old identities and use fresh new identities in their place.

Ruffing, Kate, and Schröder [RKS15] introduce "accountable assertions", which enable timelocked deposits in Bitcoin that are revoked upon evidence of double spending. Pass and Shelat [PS16] also suggest a Bitcoin-specific penalty mechanism to deter rational customers and merchants from cheating.[5] Unfortunately, both of these works do not provide an economic analysis to indicate how large a penalty should be to deter double spending. Such an analysis is crucial: how could detect-and-punish be a deterrent if double spending were to yield *unbounded* additional utility?

1.1 Our contributions

We overcome the aforementioned limitations via a combination of cryptographic and economic techniques. We adopt a "detect-and-punish" approach in which cryptography is used to retroactively detect and economically punish double spending and, separately, an economic analysis clarifies how much to punish so as to deter double spending in the first place. More precisely, we present the following three contributions.

1.1.1 Economic analysis of double spending for offline probabilistic payments

We characterize the additional utility that can be gained by double spending via offline probabilistic payments. We suppose that: (i) every probabilistic payment is backed by an advance deposit;[6] (ii) all macropayment double spends can be detected; and (iii) if a merchant detects a double spend then he reports it, and doing so results in the revocation of the cheating customer's deposit. (Our cryptographic constructions will provide suitable mechanisms for these tasks.) We then ask: *how large must the deposit be in order to deter double spending?*

We provide a simple yet powerful analysis that answers this question under reasonable network behavior. Namely, let T denote the time it takes to catch a macropayment double spend (e.g., in Bitcoin one could take T to be the network's broadcast time). Within any period of time T, let A denote the maximum cumulative value of probabilistic

[5] We also note that two of the three schemes in [PS15] do not support offline payments, and the remaining one only provides "fast online payments" where an online (publicly verifiable) trusted party assists the ledger by processing macropayments faster.

[6] One deposit may back multiple payments; in particular, an honest customer may use a single deposit to back all of his payments.

payments and W the maximum cumulative value of macropayments; our analysis will show that imposing bounds on these quantities is *necessary*. To simplify discussions, we make the assumption that *only* macropayment (and not nullpayment) double spends are detectable; our analysis extends to the case where nullpayment double spends may also be detected eventually (see Remark 1).

Below we informally state our theorem, for simplicity in the special case where the macropayment value V and the payment probability p are fixed across all probabilistic payments, and all merchants share the same detection time T. The formal statement that we prove is in fact more general, because it applies even when these quantities are chosen dynamically and arbitrarily across different payments.

Theorem 1 (informal statement of Thm. 4).

(a) If the deposit is at least W, then there is no worst-case utility gain in double spending.

(b) If the deposit is at least $(1-p)V + A$, then there is no average-case utility gain in double spending.

(c) Both bounds above are tight.

Our theorem has a simple interpretation: the required deposit amount equals the maximum financial activity that can happen within any time period of T. Namely, if macropayments have maximum total worth W within time T, the deposit must be at least W (w.r.t. worst-case utility); and if probabilistic payments have maximum total worth A within time T, the deposit must be at least $\approx A$ (w.r.t. average-case utility). Note that it is unsurprising that the two statements in the theorem depend on the two different quantities W and A, because they target different notions of utility; also note that, while one can take $pW \leq A$ without loss of generality, a bound on W does *not* always imply a bound on A (there could still be arbitrarily many probabilistic payments, though with extremely small probability).

But which of the two bounds should one use in practice? Naturally, the worst-case bound is safer than the average-case bound; however, an appropriate setting of W will be $\Omega(1/p)$ larger than A, which implies a substantial increase in the required deposit. The choice between the two depends on whether one cares about malicious customers that are lucky with even very small probability (as opposed to focusing on their average gains possibly across many deposits).

As already mentioned, bounding the value of probabilistic payments (via A) or macropayments (via W) within time T is *necessary* because our bounds are tight (i.e., there exist double-spending strategies that achieve them). In the "real world" these bounds may be imposed by the environment (e.g., limited network throughput), or the merchants (e.g., they accept up to a given number of payments within time T).

In terms of analysis, our proof shows that any additional utility gained via double spending must come from *macropayment* double spending. This may be surprising because, superficially, one may think that *nullpayment* double spending also contributes to additional utility; e.g., one may think that a malicious customer gains pV for every nullpayment double spend. This proposition is alarming: in the worst case there could be infinitely-many nullpayment double spends (which imply infinite additional utility); and in the average case there could be clever strategies that leverage double spends across

multiple merchants to lower the probability of detection. We prove that this is not the case: we use a simulation argument to show that *the naive strategy of double spending as much as possible is the best strategy* (i.e., maximizes additional utility), both in the worst case and in the average case. In particular, we learn that the best strategy always leads to detection (after a time period of T) and that additional utility *is finite even in the worst case* (if W is finite). Details of our analysis are in Section 3.

We believe our theorem to be of independent interest because it applies to virtually any (centralized or decentralized) setting that enforces a deposit mechanism for offline payments. One such setting could be probabilistic *smart contracts* (an application suggested by [PS15, PS16]). A thorough understanding of the economic benefits of double spending is necessary to ensure that such smart contracts, as well as other applications, function as intended.

Example. As a demonstration, we invoke our theorem on parameters that could fit the application of *advertisement-free content delivery*, to see what conclusions our economic analysis gives us. Suppose that we consider a Bitcoin-like setting, where (i) transaction fees are typically a few cents; and (ii) we could take the detection time T to be, e.g., 20 minutes, which is typically two blocks (ideal block generation follows an exponential distribution with a mean of 10 minutes). Suppose further that we fix the deposit to be $D := \$200$ and the expected value of the probabilistic payment to be $\$0.1$ (similar size as a transaction fee); concentration bounds then suggest that, subject to the condition $pV = \$0.1$, good choices are $V := \$10$ and $p := 1\%$. Note that these settings imply that we can take W up to $D = \$200$ and A up to $D - (1-p)V = \$190.1$. Then our theorem implies that: (1) Even the *luckiest* double spending user has no extra utility gain if the cumulative value of macropayments every 20 minutes is less than $\$200$ (that is, the number of macropayments every 20 minutes is less than 20), regardless of how much nullpayment double spending occurred. (2) A double spending user has no extra utility gain on average if the cumulative value of probabilistic payments every 20 minutes is less than $\$190.1$ (that is, the number of probabilistic payments every 20 minutes is less than 1901).

1.1.2 Decentralized anonymous micropayments

We formulate the notion of a *decentralized anonymous micropayment* (DAM) scheme. This notion formalizes the functionality and security properties of an offline probabilistic payment scheme that enables parties with access to a ledger to conduct transactions with one another, directly and privately. To realize the requirements of our economic analysis, a DAM scheme enables parties to set up deposits, which are revoked when macropayments reveal that double spending has occurred. Crucially, the security guarantees of a DAM scheme guarantee anonymity not only across macropayments but also across nullpayments, so that even the "offline stream of payments" remains unlinkable.

We construct a DAM scheme and prove its security under specific cryptographic assumptions. Our two main building blocks are decentralized anonymous payment (DAP) schemes [BCG$^+$14] and fractional message transfer schemes (see below).

Theorem 2 (informal). *Given a decentralized anonymous payment scheme and a fractional message transfer scheme (and other standard cryptographic primitives) there exists a DAM scheme.*

Formally capturing the notion of a DAM scheme and proving security of our construction was quite challenging due to the combination of rich functionality and strong anonymity guarantees. Parties can mint standard coins, deposits, or lottery tickets; they can withdraw deposits; they can pay each other with deterministic payments, switch coin types; they can also pay each other with probabilistic payments; they can revoke deposits of cheating parties — all of this while essentially revealing no information about origins, destinations, and amounts of money transfers. In particular, two features of our construction required particular attention: (1) revocation of an unknown cheating party's deposit when two macropayments with the same ticket are detected; and (2) monitoring of payment value rates (as required by our economic analysis) despite deposits being anonymous. Deterministic payments in our construction are non-interactive, while probabilistic payments consist of a 3-message protocol between a sender and a receiver; it is an interesting open question whether these can be made non-interactive as well.

We express the security of a DAM scheme via the ideal-world/real-world paradigm, specifying a suitable ideal functionality, and we prove our construction's security via a simulator against non-adaptive corruptions of parties. We consider security in the standalone setting, and leave security under composition to future work (that perhaps can build upon the work of [KMS+16]).

1.1.3 Fractional message transfer

A key ingredient in our construction of DAM schemes is *fractional message transfer* (FMT): a primitive that enables probabilistic message transmission between two parties, called the 'sender' and the 'receiver'. Informally, FMT works as follows: (i) the receiver samples a one-time key pair based on a transfer probability p; (ii) the sender uses the receiver's public key to encrypt a message m into a ciphertext c; (iii) the receiver uses the secret key to decrypt c, thereby learning m, but only with the pre-defined probability p (and otherwise learns no information about m).

We thus (1) formulate the notion of an *FMT scheme*, which formally captures the functionality and security of probabilistic message transmission, and (2) present an efficient construction that works for probabilities that are inverses of positive integers.

Theorem 3 (informal). *In the random oracle model and assuming the hardness of* DDH *in prime-order groups, there exists an FMT scheme that works for transfer probabilities $p = 1/n$ with $n \in \mathbb{N}$. Moreover, the number of group elements and scalars in the public key and ciphertext is constant (independent of n); see Table 1.*

Our definition of FMT is closely related to *non-interactive fractional oblivious transfer* (NFOT), which was studied in the context of 'translucent cryptography' as an alternative to key escrow [BM89, BR99]. Namely, prior definitions target *one-way security*, which protects *random* messages. While one-way security suffices to encapsulate random secret keys (the setting of translucent cryptography), it does not suffice for probabilistically transmitting non-random messages (as needed in our construction). Therefore, our definition of an FMT scheme targets a fractional variant of *semantic security*, which we express via two properties: *fractional hiding* and *fractional binding*. Furthermore, since in our system any party can act as both sender and receiver, we require the FMT scheme to be *composable*. Our construction achieves this via simulation-extractability.

Our construction of FMT is loosely related to the constructions in [BM89, BR99], which (like our construction) build on the Elgamal encryption scheme [Elg85]. In fact, such constructions, if analyzed under the hardness of DDH rather than CDH, are likely to yield FMT according to our stronger definition. We did not carry out such an analysis, but instead chose to construct a scheme that is more efficient than prior work for the case of $p = 1/n$ (these probabilities suffice for our application); we assume hardness of DDH and work in the random oracle model in order to take advantage of certain Σ-protocols. See Table 1 for a comparison of our construction with prior work.

scheme	security	assumption	transfer probability	size of public key		size of ciphertext		# exponentiations	
				group elts.	scalars	group elts.	scalars	to encrypt	to decrypt
[BM89]	one-way	CDH	$1/2$	2	—	2	—	2	1
[BR99, § 5.1]	one-way	CDH	$1/n$	n	—	2	—	2	1
[BR99, § 5.1]	one-way	CDH	$(n-1)/n$	n	—	2	—	2	1
[BR99, § 5.2]	one-way	CDH	a/n *	$2\log_2 n$	—	$2\log_2 n$	—	$4\log_2 n$	$2\log_2 n$
[BR99, § 5.3]	one-way	CDH	a/n	$a+n$	—	2	—	2	1
our FMT	semantic	DDH + RO	$1/n$	2	3	2	2	4	4

* n is restricted to be a power of 2.

Table 1: Comparison of prior NFOT schemes vs. our FMT scheme. All constructions assume a common random string.

1.2 Prior work on micropayment channels

Micropayment channels were introduced by Hearn and Spilman [HS12, Bit13], and further studied by Poon and Dryja [PD16] and Decker and Wattenhofer [DW15]. Roughly, a micropayment channel enables a sender and a receiver to set up a contract by way of an online (slow) transaction that escrows funds, after which the sender and receiver can update the contract, and thus the relative split of the escrowed funds, without recording the new contract on the blockchain. Thus payments can be made instantaneously. These can be dynamically combined to obtain multi-hop "payment channel networks" that go through several intermediaries, by using hashed timelock contracts; this technique amortizes the cost of setting up a new channel for new receivers. From the perspective of our work, micropayment channels have several limitations in terms of economics, functionality, and privacy.

Economic limitations of payment channels. First, payment channels in general require a channel to be established in advance with a party: payments are only instantaneous with advanced preparation. To alleviate this constraint, payment channel networks allow transactions with arbitrary new parties provided there exists a path of existing channels between the payer and payee.

Such networks have limitations. First, considerable capital is escrowed in the many pairwise channels forming the network. The capital requirements may exceed those

required for deposits in probabilistic micropayments. Both settings require escrowed funds proportional to a user's economic activity (either for the double spend deposit or the "last mile" channel between the user and the payment network), but payment channel networks escrow similar amounts in each edge of the network. Second, a variety of pressures, including minimizing the capital escrowed, may centralize such networks into a hub-and-spoke model.

Privacy limitations of payment channels. Payment channels reveal to the world that a given pair of parties have a channel between them, the opening value of that channel, and the final closing value. More importantly, especially for applications like advertisement-free content delivery, payment channels provide no privacy between the parties on the channel: if Alice pays say Wikipedia every time she views a page, then each of those views is linked to the channel she established just as effectively as if she had a tracking cookie in her browser.

Attempts to add privacy, either from intermediate nodes in the network [HAB+16] or from recipients and intermediaries [GM16], to payment channels hit some seemingly fundamental limitations of the payment channel setting. First, the anonymity set when paying a given receiver is composed only of those users who have opened channels with the receiver. This is likely far smaller than the global anonymity set provided by probabilistic payments. Moreover, the receiving party can arbitrarily reduce the anonymity set further by closing channels. This leaves open a range of attacks that are not present in a system with a global anonymity set.

Finally it is unclear if non-hub-and-spoke private payment networks are scalable or can provide privacy for payment values from intermediary nodes in the network. When a payment is made via two intermediaries (i.e. $A \to I_1 \to I_2 \to B$), some combination of I_1 and I_2 must know the balance of their pairwise channel at any given time or they could not close the channel. Thus the value of any payment relayed through multiple parties cannot be completely private. Moreover, discovering a multi-hop route between two parties in a diverse and large network without leaking any identifying information seems costly at scale. While [GM16] extend their point-to-point channel protocol to a hub-and-spoke model that alleviates both these concerns, such a network is inherently centralized.

2 Techniques

We discuss the intuition and techniques behind our results, first for our cryptographic construction (Section 2.1) and then for our economic analysis of double spending (Section 2.2).

2.1 Constructing decentralized anonymous payments

We discuss our design of a decentralized anonymous micropayment (DAM) scheme via a sequence of candidate constructions, each fixing problems of the previous one; the last one is a sketch of our construction.

2.1.1 Attempt 1: non-anonymous probabilistic payments + DAP

We begin with a natural candidate construction for a DAM scheme. The idea is to combine two primitives, one providing probabilistic payments and the other anonymity. For example, consider: (1) the scheme `MICROPAY1` of [PS15], which provides probabilistic

payments for Bitcoin; and (2) a *decentralized anonymous payment* (DAP) scheme [BCG+14], which provides privacy-preserving payments for Bitcoin-like currencies.

To make `MICROPAY1` privacy-preserving, we could try to replace its Bitcoin payments with DAP payments, which hide the payment's origin, destination, and amount. Thus, when a probabilistic payment goes through, and the corresponding DAP (macro-)payment is broadcast, others cannot learn this information about the payment. However, this idea does *not* provide the strong anonymity guarantees that we seek, as we now explain.

Problem: not fully anonymous. Despite the anonymity guarantees provided by the DAP scheme, merchants still learn information about their customers' spending habits. Each DAP payment includes a unique serial number corresponding to the underlying "coin" that was spent by that payment; this is used to prevent double spending of DAP coins. In the above proposal, the customer sends the merchant this serial number regardless of whether the payment becomes a nullpayment or a macropayment. Since the same underlying DAP payment and serial number are used across multiple probabilistic payments (to amortize fees), this compromises customer anonymity because a merchant learns (1) which (macro or null) payments to him were made with the same escrow; and (2) which macropayments to other merchants were made with an escrow used for payments to him. This compromise in anonymity gets even worse if merchants share such information with one another.

Moreover, recall (from Section 1.1) that it is not possible to prevent double spending in the setting of offline probabilistic payments. Pass and Shelat note this in the full version of their paper [PS16], and propose adding a 'penalty escrow' to the scheme `MICROPAY1`; the escrow is burned upon evidence of double spending. But observe that anonymity for penalty escrows poses a similar challenge: to prove that a penalty escrow is unspent, a merchant reveals its serial number, once again enabling merchants to link probabilistic payments by learning about their escrows.

Overall, while the above ideas do achieve unlinkability of macropayments, customers have little meaningful privacy until nullpayments and escrows are also unlinkable.

2.1.2 Attempt 2: commit to DAP payment + probabilistic opening + private deposit coins

One way to address the anonymity problems of the previous attempt is to ensure that the merchant learns the serial number only when the payment turns into a macropayment (and, conversely, learns nothing otherwise). Then, to enable the aforementioned penalty escrow mechanism, a customer creates a special 'deposit' coin.

Then, the modified protocol works as follows: (1) the customer sends to the merchant a commitment to a DAP payment and to a 2-out-of-n share of the deposit serial number; (2) the customer and merchant engage in a protocol that opens the commitment with probability p (opening thus corresponds to a macropayment, and not opening corresponds to a nullpayment); (3) when publishing a macropayment to the ledger, the merchant also publishes the secret share.

The probabilistic opening hides the serial number of the coin in the DAP payment until a macropayment occurs, and the secret share hides the deposit serial number until a macropayment double spend occurs. To punish a double spending customer, the merchant obtains (from the network or from the ledger) two secret shares of the deposit serial

number from two macropayments and reconstructs the serial number. He then publishes this to the ledger, thereby blacklisting the deposit.

One issue that must be addressed is ensuring that the secret shared deposit serial number corresponds to a valid deposit. To do this, first notice that there are two kinds of blacklisted deposits: those whose serial number appears on the ledger (in previous 'punish' transactions), and those that have been revoked in the current epoch. The serial numbers of the latter kind are broadcast across the network, but have not yet appeared on the ledger.

To prevent users from using blacklisted deposits of the first kind, a customer must prove to the merchant that his deposit's serial number does not appear on the ledger (this can be done efficiently [MRK03]). To prevent use of deposits of the second kind, customers must also send to the merchant a tag derived from the deposit's serial number. Since anyone with access to this serial number can compute this tag, merchants can deduce if a deposit has been revoked by checking if this tag has been computed with a blacklisted deposit's serial number. The customer accompanies the tag with a zero-knowledge proof that the deposit used for this tag is consistent with the share inside the commitment.

The aforementioned proposal, however, is still vulnerable to attacks.

Problem: front-running deposit revocation. While deposits are intended to deter double spending, customers may try to withdraw a deposit before it is blacklisted, thereby rendering punishment ineffective.

Problem: merchant aborts. At the end of the commitment opening protocol, the merchant can refuse to inform the customer of whether or not the commitment was opened. This poses a problem for the customer because if the commitment was in fact opened, the merchant has learned the serial number and a share of the deposit, enabling him to: (i) track the customer and learn when they spend the coin with another merchant, and (ii) revoke the customer's deposit after the (honest) customer next spends the coin, with another merchant or the same one.

2.1.3 Outline of our construction

The deposit mechanism described so far is insufficient to deter double spending. The problem is that there is no restriction on how and when coins used for probabilistic payments and for deposits can be transferred; in particular, a cheating customer can double spend these back to himself while at the same time engaging in a probabilistic payment with a merchant. We address this problem by (i) partitioning coins into different types depending on their different uses, and (ii) restricting transfers between coins depending on their types. We now outline how we carry out this plan.

First, we extend the notion of a DAP scheme to allow users to associate public and private information strings when minting a coin. Users can now store a coin's type in its public information string, and we allow three types of coins: in addition to the "standard" coin type, we introduce deposits and tickets. A ticket is bound to a deposit by storing the deposit inside the ticket's private information string. We thus have the following semantics:

– *Coins* are used for deterministic DAP payments (whose processing fees are not amortized).

– *Deposits* are used to back tickets and are revoked when two macropayments using the same ticket are detected.
– *Tickets* are used for probabilistic payments; every ticket is bound to a single deposit at minting time, and can be spent provided that the associated deposit is valid (i.e., has not be transferred to a coin, or revoked).

We also restrict the set of possible transactions depending on the types of coins involved, as follows.

– *Transactions with coins:* Coins can be used to create other coins, deposits, or tickets. In particular, coin-to-coin transactions preserve the deterministic payment functionality of the underlying DAP scheme.
– *Transactions with deposits:* Deposit-to-coin transactions let customers withdraw deposits, though not immediately, since these transactions become active only after an *activation delay* Δ_{w} that is a parameter of the system.
– *Transactions with tickets:* Ticket-to-coin transactions enable probabilistic payments; they are associated with a secret share of the ticket's deposit and with a deposit-derived tag that allows merchants to detect the validity of the ticket's deposit. Ticket-to-ticket transactions omit the secret share and tag and (like deposit-to-coin transactions) become active only after an *activation delay* Δ_{r} that is a parameter of the system.

Restrictions on inter-type transactions are achieved via a *pour predicate* that checks that input and output coin types satisfy the above restrictions. Having made these modifications, we can now resolve the issues of the previous proposal.

Preventing deposit theft. Deposit-to-coin transactions now have a delayed activation, so customers can no longer withdraw deposits before they are blacklisted, as merchants have enough time to post deposit revocations to the ledger.

Recovering from merchant aborts. Since we cannot know what is the utility gain of a merchant for learning about the spending patterns of a customer, we cannot effectively deter merchant aborts by economic means. Instead, at the end of our commitment opening protocol, we require the merchant to prove to the customer whether or not he could open the commitment. If the merchant fails to do so, we allow customers to "refresh" their tickets by creating a ticket-to-ticket payment to themselves. Since the new ticket has a different serial number that merchants have not yet seen, they cannot track the new ticket's transaction history. Finally, since ticket-to-ticket transactions become active only after a delay, the new tickets cannot be spent immediately, thus allowing merchants to post macropayments over the old ticket.

The above sketch omits many technical details, including how a DAM scheme interacts with the economic analysis. See the full version.

2.2 Intuition for our economic analysis of double spending

Our economic analysis characterizes the additional utility that customers can gain by double spending in offline probabilistic payments. We discuss the intuition for the analysis via an example; details of the analysis are in Section 3 (the formal statement is Theorem 4). Recall that we assume that: (i) every probabilistic payment is backed by an advance deposit, and (ii) macropayment double spends are detected within time T, and result in deposit revocation.

At a high level, the deposit must be at least as large as the additional utility that a malicious customer gains by double spending until that deposit is revoked; additional utility occurs when the customer double spends, and accumulates until cheating is detected and every merchant has blacklisted the customer. If we can bound the value of payments in this period of time, then we can derive a corresponding bound on the additional utility gained, and thus bound the deposit.

A naive analysis, however, yields an impractically large bound, because the natural definition of "additional utility" is too coarse. We illustrate this issue via an example: a malicious customer $\tilde{\mathsf{C}}$ selects two merchants $\mathsf{M}_1, \mathsf{M}_2$, and uses the same "lottery ticket" to conduct parallel probabilistic payments $\widetilde{\mathsf{pay}}_1, \widetilde{\mathsf{pay}}_2$ to $\mathsf{M}_1, \mathsf{M}_2$ respectively. The merchants cannot immediately detect that $\tilde{\mathsf{C}}$ is cheating because $\tilde{\mathsf{C}}$ is indistinguishable from an honest user so far. If both $\widetilde{\mathsf{pay}}_1$ and $\widetilde{\mathsf{pay}}_2$ become macropayments, which happens with probability p^2, then the merchants (eventually) catch $\tilde{\mathsf{C}}$ cheating, and revoke $\tilde{\mathsf{C}}$'s deposit of value D. Consider the following two analyses.

(i) A naive analysis. The malicious customer $\tilde{\mathsf{C}}$ earns an additional utility of pV compared to an honest customer, and is caught and punished by D with probability p^2. Hence, to deter $\tilde{\mathsf{C}}$ from cheating, the deposit amount should be such that $p^2 D > pV$, which is equivalent to $D > V/p$.

(ii) A better analysis. The average-case utility $\mathbb{E}[\mathcal{U}(\mathsf{C})]$ of an honest customer C for any probabilistic payment is zero: C gains pV with probability $1-p$, and $pV - V$ with probability p. Instead, the utility $\mathcal{U}(\tilde{\mathsf{C}})$ of the malicious customer $\tilde{\mathsf{C}}$ has four cases, as given in Table 2; also, $\tilde{\mathsf{C}}$ is caught and punished by D with probability p^2. Thus, the deposit amount should be such that $p^2 D > \mathbb{E}[\mathcal{U}(\tilde{\mathsf{C}})] = 2pV - (1-(1-p)^2)V$, which is equivalent to $D > V$.

$\widetilde{\mathsf{pay}}_1$ \ $\widetilde{\mathsf{pay}}_2$	null	macro
null	$2pV$	$2pV - V$
macro	$2pV - V$	$2pV - V$

Table 2: Utility $\mathcal{U}(\tilde{\mathsf{C}})$ of the malicious customer $\tilde{\mathsf{C}}$.

pay_1 \ pay_2	null	macro
null	$2pV$	$2pV - V$
macro	$2pV - V$	$2pV - 2V$

Table 3: Utility $\mathcal{U}(\mathsf{C})$ of the honest customer C.

How do the two analyses differ? The first analysis states that the deposit amount D must be greater than V/p while the second states that it must be greater than V, which is a much smaller lower bound. This is because the first analysis adopted an intuitive, but coarse, definition of additional utility, which did not consider the fact that a malicious customer does not gain additional utility unless two macropayments with the same ticket occur. Indeed, the utility $\mathcal{U}(\mathsf{C})$ of an honest user C that uses two *different* tickets to make two parallel probabilistic payments $\mathsf{pay}_1, \mathsf{pay}_2$ is in Table 3. By comparing $\mathcal{U}(\tilde{\mathsf{C}})$ and $\mathcal{U}(\mathsf{C})$, one can see that the utility function differs *only when two macropayments occur*, where, if there is no deposit/punishment, $\tilde{\mathsf{C}}$ gains extra utility of V by paying only one macropayment instead of paying two as C does. In sum, any additional utility gained via double spending *must come from macropayment double spends*.

Towards a general analysis. The above discussion suggests that the additional utility of $\tilde{\mathsf{C}}$, which we denote by $\mathcal{U}'(\tilde{\mathsf{C}})$, should be defined as follows:

$$\mathcal{U}'(\tilde{\mathsf{C}}) := \begin{cases} V \text{ if } \widetilde{\mathsf{pay}}_1, \widetilde{\mathsf{pay}}_2 \text{ are macropayments} \\ 0 \text{ otherwise} \end{cases} .$$

More generally, the additional utility of any malicious customer $\tilde{\mathsf{C}}$ is the extra gain compared to an honest customer achieving the same outcome. This can be computed by considering an honest customer C that simulates the behavior of $\tilde{\mathsf{C}}$ while only using unspent tickets; the extra gain arises from the fact that C has "paid" for these other unspent tickets while $\tilde{\mathsf{C}}$ has not. By understanding the maximum of this refined notion of additional utility we can derive the minimum amount of deposit needed such that, for any double spending attack, there is a non-double-spending strategy that achieves better utility, in the worst-case and in the average-case respectively. See Section 3 for a formal argument of this intuition, as well as a discussion of the implications of our economic analysis.

3 Economic analysis of double spending for offline probabilistic payments

We provide the economic analysis that characterizes the additional utility that can be gained by double spending via offline probabilistic payments. This section is organized as follows. First, we informally describe dynamics that model offline probabilistic payments (Section 3.1). Then, we define a formal game that captures these dynamics and analyze this game (Section 3.2). Finally, we discuss the interpretation and consequences of our economic analysis (Section 3.3).

3.1 Informal description of payment dynamics

We informally describe the dynamics of arbitrary probabilistic payments from customers to merchants. A concrete example is the setting of *advertisement-free Internet*: a customer is a user surfing the Internet; a merchant is a web server; every HTTP request by a user to a web server is accompanied by a probabilistic payment from that user to the web server (to buy an ad-free HTTP response).

Abstraction of probabilistic payments. A probabilistic payment is an interactive protocol between a customer and a merchant. The customer's input is a *ticket* $\mathbf{t} = (\mathsf{t}, p, V, \mathbf{d})$ where $\mathsf{t} \in \{0,1\}^*$ is the unique *ticket identifier*, $p \in [0,1]$ is the *payment probability*, $V \in \mathbb{R}_{\geq 0}$ is the *macropayment value*, and $\mathbf{d} = (\mathsf{d}, D)$ is the *deposit*, which consists of a unique *deposit identifier* $\mathsf{d} \in \{0,1\}^*$ and a *deposit value* $D \in \mathbb{R}_{\geq 0}$. Informally, the customer first convinces the merchant that the deposit is not "invalid", and then the customer pays V to the merchant with probability p. The two outcomes are called a *nullpayment* and a *macropayment*, and involve different protocol outputs.

Detectable double spends. At any moment in time, a deposit is in one of two states: *valid* or *invalid*. Each deposit is initially valid. When two macropayments occur on the same ticket $\mathbf{t}$, the associated deposit $\mathbf{d}$ becomes invalid, once and for all. We call this event a *macropayment double spend*, and we assume that, in this case, the underlying probabilistic payment protocol enables merchants to eventually learn that $\mathbf{d}$ (more

precisely, its identifier) has become invalid;[7] we denote by T_{M} the time for merchant M to learn this from the moment the macropayment double spend occurred. The fact that $\max_{\mathsf{M}} T_{\mathsf{M}} > 0$ is the fundamental reason that allows a malicious customer to gain any additional utility.

Finally, we make the simplifying assumption that, while macropayment double spends are detectable, nullpayment double spends are undetectable. Our analysis does extend to the case where (not necessarily all) nullpayment double spends are also detectable; see Remark 1.

Honesty of merchants. We assume that *merchants behave honestly*. Thus, every merchant (a) rejects aborted payments (e.g., due to invalid deposits); (b) honors successful payments (e.g., replies with an ad-free HTTP response) regardless of whether the payment resulted in a nullpayment or macropayment; (c) reports detected double spends; more generally, (d) follows the probabilistic payment protocol (e.g., uses fresh randomness in each instance of the protocol, broadcasts any messages to all other merchants as instructed, and so on).

In principle, merchants may deviate from the aforementioned honest behavior in a variety of ways. For instance, a merchant may "honor" an aborted payment (e.g., regardless of the validity of the customer's deposit); or the merchant may not honor a successful payment (e.g., does not reply to the HTTP request); or the merchant may abort and prevent the customer from learning the payment's outcome; or the merchant may not report a detected double spend.

However, we assume that all merchants behave honestly because the only incentive for a merchant to deviate comes from colluding with malicious customers, and we cannot prevent such collusions. Indeed, if a merchant does not collude with any malicious customer, then for the merchant it is individually rational to behave honestly, because: (i) some malicious merchant behavior (e.g., "honoring" an aborted payment, or using correlated randomness across payments) does not increase the merchant's utility; (ii) other malicious merchant behavior (e.g., not honoring a successful payment) decreases the customer's utility, but taking into account this possibility does not affect a customer's maximum additional utility (the quantity we study) and ruling it out significantly simplifies the analysis. However, a malicious customer could convince a merchant to *not* report a double spend by offering side payments as compensation; if the merchant has already replied to the customer's payment then this collusion may indeed be economically attractive, but we cannot systematically prevent such side payments in all applications. (In the setting of micropayments, V is small so a merchant may prefer to see the malicious customer punished, after losing V, rather than receiving compensation.)

Honest vs. malicious customers. Our goal is to characterize the additional utility obtained by any malicious customer, when compared to what is possible by honest customers. We now discuss both kinds of customers.

Honest customers. For an honest customer, a ticket $\mathbf{t}$ is in one of three states: it is *spent* if a probabilistic payment on it has resulted in a macropayment; otherwise, it is

[7] Exactly *how* merchants learn $\mathbf{d}$'s identifier depends on the details of a construction, and is orthogonal to our economic analysis; ditto for exactly how the monetary funds escrowed in $\mathbf{d}$ are revoked after $\mathbf{d}$ becomes invalid.

occupied if it is being used in a probabilistic payment; otherwise, it is *unspent* (i.e., it never resulted in a macropayment, nor is it being used in a probabilistic payment).

At any moment in time, an honest customer may select any number of merchants, and initiate any number of probabilistic payments in parallel to every one of them. Each probabilistic payment uses a distinct unspent ticket, which immediately becomes occupied, and at the end of the payment protocol becomes either unspent or spent. The selected tickets may or may not have different deposits that back them; deposits are never invalidated for honest customers. In sum, an honest customer maintains the invariant that an occupied ticket does not participate in more than one payment at a time, and a spent ticket does not participate in future payments.

Malicious customers. A malicious customer may deviate from the aforementioned honest behavior in a variety of ways, as we now describe. Like an honest customer, a malicious customer owns an arbitrary number of tickets and deposits; unlike an honest customer, a malicious customer may use an occupied ticket in multiple payments, or may use a spent ticket in future payments (hence, a ticket of a malicious customer could be in *both spent and occupied states at the same time*). We give some examples of malicious behavior.

- **One-ticket-one-merchant attack.** A malicious customer $\tilde{\mathsf{C}}$ has a ticket $\mathbf{t}$ and selects a merchant M; then $\tilde{\mathsf{C}}$ initiates multiple probabilistic payments to M in parallel, and continues using the same ticket $\mathbf{t}$ even after it is spent. The merchant M cannot detect that $\tilde{\mathsf{C}}$ is cheating until M receives two macropayments relative to the same ticket $\mathbf{t}$.
- **One-ticket-multiple-merchant attack.** A malicious customer $\tilde{\mathsf{C}}$ has a ticket $\mathbf{t}$ and selects two merchants $\mathsf{M}_1, \mathsf{M}_2$; then $\tilde{\mathsf{C}}$ conducts a sequence of probabilistic payments to M_1, using $\mathbf{t}$ until it is spent to M_1. In parallel, $\tilde{\mathsf{C}}$ adopts the same strategy with M_2, until $\mathbf{t}$ is spent to M_2. Observe that $\tilde{\mathsf{C}}$ acts like an honest customer to M_1 and M_2 individually; hence, the two merchants cannot detect that $\tilde{\mathsf{C}}$ is cheating until they communicate.
- **Multiple-ticket-multiple-merchant attack.** More generally, a malicious customer $\tilde{\mathsf{C}}$ has multiple tickets $\mathbf{t}_1, \mathbf{t}_2, \ldots$ and selects multiple merchants $\mathsf{M}_1, \mathsf{M}_2, \ldots$; then $\tilde{\mathsf{C}}$ conducts a sequence of probabilistic payments to M_1, using $\mathbf{t}_1$ until it is spent to M_1. Then $\tilde{\mathsf{C}}$ switches to $\mathbf{t}_2$ and continues making probabilistic payments to M_1 until $\mathbf{t}_2$ is spent. The customer $\tilde{\mathsf{C}}$ continues in this way until all the tickets are spent to M_1. In parallel, $\tilde{\mathsf{C}}$ adopts the same strategy with every other merchant. Observe again that $\tilde{\mathsf{C}}$ acts like an honest customer to each merchant individually; hence, the merchants cannot detect that $\tilde{\mathsf{C}}$ is cheating until they communicate.

Recall that, no matter what a malicious customer does, whenever two macropayments relative to the same ticket $\mathbf{t}$ occur, the deposit of $\mathbf{t}$ becomes invalid, and eventually (after at most time $\max_{\mathsf{M}} T_{\mathsf{M}}$) all merchants learn about this.

Towards a formal game. The above discussion leads us to the following informal description of arbitrary dynamics of probabilistic payments from a potentially-malicious customer to honest merchants; this description is only an intermediate step that we provide for intuition, because we formally define an abstract game in Section 3.2 below.

For each time t, let $\mathcal{I}(t)$ denote the set of deposit identifiers of invalid deposits at time t. This set is not maintained by anyone: by definition it contains the correct identifiers at any time. It is public and, hence, known to the customer.

Suppose that a customer initiates a probabilistic payment with merchant M at time t, using a ticket $\mathtt{t} = (\mathtt{t}, V, p, (\mathtt{d}, D))$. If $\mathtt{d} \in \mathcal{I}(t - T_{\mathsf{M}})$ (the deposit identifier belongs to an invalid deposit) then the payment aborts. Otherwise, (i) with probability $1 - p$, both parties receive the output `null`; (ii) with probability p, both parties receive the output `macro`.

Crucially, the decision of whether a payment aborts depends only on the global information from T_{M} units of time "into the past", because, in the worst case, there is a delay of T_{M} for merchant M to learn that a deposit has been invalidated. Of course, the merchant M may happen to learn this information faster than that; though modeling this fact does not ultimately change the maximum additional utility, so we ignore this for simplicity. This means that all merchants "behave the same" and thus we replace them with a single abstract player, 'Nature', in the next section.

Note that a construction of a probabilistic payment should also involve a check of whether the deposit value D is "large enough" to back the payment (as informed by our economic analysis). We ignore this check (and how it can be performed) because it is irrelevant to the economic analysis.

3.2 The game and its analysis

We define a single-player game against Nature that captures the dynamics described in Section 3.1, namely, the dynamics of a customer $\tilde{\mathsf{C}}$ conducting arbitrary probabilistic payments with all merchants. We prove tight bounds on $\tilde{\mathsf{C}}$'s additional utility, in the worst case and in the average case. Note that, due to the additive nature of utility, we only need to analyze $\tilde{\mathsf{C}}$'s additional utility *per deposit*; hence, we restrict $\tilde{\mathsf{C}}$ to backing all his probabilistic payments with a single deposit.

As mentioned in Section 1.1.1, our analysis involves two parameters A and W, which denote the (per-deposit) maximum value of probabilistic payments and of macropayments, within any "detection time period". More precisely, let T_{M} denote the time for a merchant M to detect a detectable double spend, and let $\mathfrak{a}_{\mathsf{M}}(t)$ be the (cumulative) value of probabilistic payments accepted by M within the time period $[t, t + T_{\mathsf{M}}]$; similarly, let $\mathfrak{w}_{\mathsf{M}}(t)$ be the (cumulative) value of macropayments accepted by M within the time period $[t, t + T_{\mathsf{M}}]$. The parameters A and W are defined as $\max_t \sum_{\mathsf{M}} \mathfrak{a}_{\mathsf{M}}(t)$ and $\max_t \sum_{\mathsf{M}} \mathfrak{w}_{\mathsf{M}}(t)$ respectively. We defer to Section 3.3 a discussion of the interpretation of these parameters, and for now we focus on analyzing the additional utility in terms of these.

We argue that it suffices to study $\tilde{\mathsf{C}}$'s additional utility across merchants within a certain time period, and to consider only probabilistic payments that use spent tickets.

- *Starting point.* It suffices to analyze $\tilde{\mathsf{C}}$'s additional utility from the first time when two macropayments occur relative to the same ticket; denote by $\widetilde{\mathsf{pay}}$ the payment among these that terminates later (if they terminate simultaneously then break ties arbitrarily). Indeed, recall that $\tilde{\mathsf{C}}$'s additional utility is the extra gain compared to any honest customer achieving the same outcome. So consider the honest customer C that uses unspent tickets for every probabilistic payment that terminates before $\widetilde{\mathsf{pay}}$ does: the utilities up to then for $\tilde{\mathsf{C}}$ and C are the same. Thus, we only need to consider $\tilde{\mathsf{C}}$'s additional utility from when $\widetilde{\mathsf{pay}}$ terminates.
- *Ending point.* It suffices to analyze $\tilde{\mathsf{C}}$'s additional utility from when $\widetilde{\mathsf{pay}}$ terminates until when every merchant M has detected $\tilde{\mathsf{C}}$'s cheating. Indeed, $\widetilde{\mathsf{pay}}$ is a detectable

double spend, so within time T_{M} merchant M detects $\tilde{\mathsf{C}}$'s cheating (i.e., has learned that $\tilde{\mathsf{C}}$'s deposit is invalid) and will not accept $\tilde{\mathsf{C}}$'s probabilistic payment anymore. Moreover, $\tilde{\mathsf{C}}$'s deposit is eventually revoked.

– *Which payments.* It suffices to consider every probabilistic payment that terminates within the aforementioned time period and uses a ticket that is spent before the termination of that payment (if multiple payments terminate simultaneously then pick an arbitrary termination order for them). Throughout this section we say that these probabilistic payments *use spent tickets*, and say that the other probabilistic payments *use unspent tickets*. Indeed, consider again the honest customer C that uses unspent tickets for every probabilistic payment: the utilities for $\tilde{\mathsf{C}}$ and C are the same on probabilistic payments that use unspent tickets.

In conclusion, we only need to worry about $\tilde{\mathsf{C}}$'s additional utility from when $\widetilde{\mathsf{pay}}$ terminates until when every merchant has detected $\tilde{\mathsf{C}}$'s cheating, and it suffices to consider only probabilistic payments that use spent tickets.

Suppose that during this time period $\tilde{\mathsf{C}}$ has finished $C+1$ probabilistic payments, including $\widetilde{\mathsf{pay}}$, using spent tickets: $\widetilde{\mathsf{pay}}$ is fixed to be a macropayment, while the remaining C payments are probabilistic (i.e., turn into nullpayments or macropayments with the appropriate probability). Perhaps $\tilde{\mathsf{C}}$ only made $C+1$ payments, or perhaps the merchants accepted only the first $C+1$ and rejected the rest due to invalid or insufficient deposit. (We assume $C < \infty$ for ease of exposition, but we could replace C with ∞ and our analysis would still hold.) Either way, note that $\tilde{\mathsf{C}}$ may select the payment probability and macropayment value of a probabilistic payment based on the outcomes of prior probabilistic payments. Below we define a game that captures these payments.

Definition 1. *Consider the following single-player game against Nature.*

– *The set of randomness choices is* $[0,1]^C$*; Nature samples* $\boldsymbol{\lambda}$ *uniformly at random from* $[0,1]^C$*. We denote by* $\boldsymbol{\lambda}_{<i}$ *the first* $(i-1)$ *coordinates of* $\boldsymbol{\lambda}$ *(and define* $\boldsymbol{\lambda}_{<0}$ *and* $\boldsymbol{\lambda}_{<1}$ *to be the empty string).*

– *The player strategies* Σ *consist of tuples* $\boldsymbol{\sigma} = (p_i, V_i)_{i=0}^{C}$ *consisting of computable functions that, based on Nature's randomness choice, output parameters for all the probabilistic payments. More precisely, for each* i*,* $p_i(\boldsymbol{\lambda}_{<i}) \in [0,1]$ *is the payment probability of the* i*-th probabilistic payment, and* $V_i(\boldsymbol{\lambda}_{<i}) \in \mathbb{R}_{\geq 0}$ *is its macropayment value.*

The game proceeds as follows. The player selects a strategy $\boldsymbol{\sigma} \in \Sigma$*; afterwards, Nature samples* $\boldsymbol{\lambda}$*, whose coordinates are revealed to the player round by round. More precisely, the game is played in rounds, as follows: in round* i*, the player learns* $\boldsymbol{\lambda}_{<i}$*, and conducts a probabilistic payment (using a spent ticket) with payment probability* $p_i(\boldsymbol{\lambda}_{<i})$ *and macropayment value* $V_i(\boldsymbol{\lambda}_{<i})$*. The outcome of the* i*-th round is given by the indicator* $\mathbb{I}[\lambda_i \leq p_i(\boldsymbol{\lambda}_{<i})]$*, stating whether the payment resulting in a macropayment (the indicator equals* 1*) or nullpayment (the indicator equals* 0*).*

Observe that *all* strategies in the above game are double-spending strategies: as discussed, it suffices to consider only probabilistic payments that use spent tickets. We now turn to define additional utility. Comparing an honest customer with a malicious one,

we observe that any additional utility comes only from macropayments that involve spent tickets. More precisely, the first such macropayment (which is $\widetilde{\mathsf{pay}}$) contributes additional utility V_0 and, after that, if the i-th probabilistic payment results in a macropayment then additional utility increases by $V_i(\boldsymbol{\lambda}_{<i})$. As for nullpayments, neither an honest nor a malicious customer loses tickets, hence additional utility does not increase. Therefore, we define additional utility as follows.

Definition 2. *The* **additional utility** *of a strategy $\boldsymbol{\sigma} \in \Sigma$ on randomness $\boldsymbol{\lambda} \in [0,1]^C$ is*

$$\mathcal{U}'_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) := V_0 + \sum_{i=1}^{C} \mathbb{I}\left[\lambda_i \leq p_i(\boldsymbol{\lambda}_{<i})\right] V_i(\boldsymbol{\lambda}_{<i}) \ .$$

(Additional utility is a random variable, as it depends on Nature's randomness $\boldsymbol{\lambda}$, which is a random variable.)

We analyze the *maximum* additional utility achievable by any strategy, in the worst case and in the average case, for the game from Definition 1; these maximum values bound from below the required deposit value D (for the goal of deterring double spending). Below we define two subsets of strategies in which the bounds A or W are respected. (Note that if $C < \infty$, then $\left(\min\{p_i\}_{i=0}^{C}\right) \cdot W \leq A$ so that if A is bounded then so is W.)

Definition 3. *We define the following two sets of strategies, which respectively capture the condition that the total worth of probabilistic payments is at most A and the total worth of macropayments is most W:*

$$\Sigma_A^{\mathsf{pp}} := \left\{ \boldsymbol{\sigma} \in \Sigma : \forall\, \boldsymbol{\lambda},\ p_0 V_0 + \sum_{i=1}^{C} p_i(\boldsymbol{\lambda}_{<i}) V_i(\boldsymbol{\lambda}_{<i}) \leq A \right\} \ ,$$

$$\Sigma_W^{\mathsf{mp}} := \left\{ \boldsymbol{\sigma} \in \Sigma : \forall\, \boldsymbol{\lambda},\ V_0 + \sum_{i=1}^{C} \mathbb{I}\left[\lambda_i \leq p_i(\boldsymbol{\lambda}_{<i})\right] V_i(\boldsymbol{\lambda}_{<i}) \leq W \right\} \ .$$

We now state and prove our worst-case and average-case bounds on additional utility. (Recall that, by Yao's minimax principle, it suffices to consider only deterministic strategies [Yao77], and thus we ignore randomized ones.)

Theorem 4 (formal statement of Thm. 1). *For the game described above, the following holds.*

(a) WORST CASE: *for every randomness choice $\boldsymbol{\lambda} \in [0,1]^C$ and strategy $\boldsymbol{\sigma} \in \Sigma_W^{\mathsf{mp}}$, it holds that $\mathcal{U}'_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) \leq W$.*
(b) AVERAGE CASE: *for every strategy $\boldsymbol{\sigma} \in \Sigma_A^{\mathsf{pp}}$, it holds that $\mathbb{E}_{\boldsymbol{\lambda}}\left[\mathcal{U}'_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})\right] \leq (1 - p_0)V_0 + A$.*
(c) Both bounds are tight.

Proof. We prove the three statements in order.

Part (a). By definition of Σ_A^{pp} (see Definition 3), for every randomness choice $\boldsymbol{\lambda} \in [0,1]^C$ and strategy $\boldsymbol{\sigma} \in \Sigma_W^{\mathsf{mp}}$, it holds that $V_0 + \sum_{i=1}^{C} \mathbb{I}\left[\lambda_i \leq p_i(\boldsymbol{\lambda}_{<i})\right] V_i(\boldsymbol{\lambda}_{<i}) \leq W$;

but the quantity on the left-hand side of the inequality is $\mathcal{U}'_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})$ (see Definition 2), and the claimed statement follows.

Part (b). Recall that Nature samples $\boldsymbol{\lambda}$ uniformly at random from $[0,1]^C$, so the coordinates of $\boldsymbol{\lambda}$ are independent from one another. Therefore, for every strategy $\boldsymbol{\sigma} \in \Sigma_A^{\mathsf{pp}}$,

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\lambda}}\left[\mathcal{U}'_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})\right] &= V_0 + \mathbb{E}_{\boldsymbol{\lambda}}\left[\sum_{i=1}^{C} \mathbb{I}\left[\lambda_i \le p_i(\boldsymbol{\lambda}_{<i})\right] V_i(\boldsymbol{\lambda}_{<i})\right] \\
&= V_0 + \mathbb{E}_{\lambda_1} \cdots \mathbb{E}_{\lambda_C}\left[\sum_{i=1}^{C} \mathbb{I}\left[\lambda_i \le p_i(\boldsymbol{\lambda}_{<i})\right] V_i(\boldsymbol{\lambda}_{<i})\right] && \text{(by independence)} \\
&= V_0 + \sum_{i=1}^{C} \mathbb{E}_{\boldsymbol{\lambda}_{<i}}\left[p_i(\boldsymbol{\lambda}_{<i}) V_i(\boldsymbol{\lambda}_{<i})\right] \\
&\le (1-p_0)\, V_0 + A. && \text{(by definition of } \Sigma_A^{\mathsf{pp}})
\end{aligned}$$

as claimed.

Part (c). Consider the following two strategies consisting of a single probabilistic payment after $\widetilde{\mathsf{pay}}$ (of value V_0):

- Choose $\boldsymbol{\sigma}$ such that $C := 1$, $p_1 := 1$, and $V_1 := W - V_0$. Note that $\boldsymbol{\sigma} \in \Sigma_W^{\mathsf{mp}}$ and, for every randomness choice $\boldsymbol{\lambda} \in [0,1]^C$, it holds that $\mathcal{U}'_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) = W$.
- Choose $\boldsymbol{\sigma}$ such that $C := 1$, $p_1 := 1$, and $V_1 := A - p_0 V_0$. Note that $\boldsymbol{\sigma} \in \Sigma_A^{\mathsf{pp}}$ and $\mathbb{E}_{\boldsymbol{\lambda}}\left[\mathcal{U}'_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})\right] = (1-p_0)\, V_0 + A$.

In sum, the first strategy shows that our worst-case bound is tight, while the second strategy shows that our average-case bound is tight.

Remark 1 (detectable nullpayment double spends). So far our analysis assumes that macropayment double spends are detectable, but nullpayment double spends are not. What if some nullpayment double spends *are* detectable? For example, merchants could maintain a partial order of all payments via a synchronous clock that ticks every second, even if the broadcast time is 10 seconds; this partial order would give chronological information on some nullpayment vs. macropayment pairs. But does such a stronger detection guarantee improve the economic bounds?

Our analysis *does* extend to this setting, and the answer is yes, but not by much. First, if some nullpayment double spends are also detectable, the additional utility of a malicious customer can only go down, so the upper bounds of our theorem continue to hold. However, the upper bounds are not tight; nevertheless, below we sketch modifications to our analysis that do recover a tight result.

- *Starting point:* the first time a detectable double spend occurs, i.e., a macropayment *or detectable nullpayment* occurs after another macropayment on the same ticket.
- *Ending point:* every merchant has detected that double spend.
- *Additional utility:* if the starting point is a macropayment double spend, the additional utility is the same, but if the starting point is a detectable nullpayment double spend, the additional utility goes down by V_0.

The rest of the analysis follows, for parameters A and W that are now defined for this new time interval. The only difference is in the initial cost of detection, due to different detection guarantees. Afterwards, only macropayment double spends provide additional utility, which are detectable in both settings. Overall, even if we had the stronger guarantee of detecting *all* nullpayment double spends, it would only save V_0 in the average-case bound.

3.3 Interpreting the payment value rates

Our analysis in Section 3.2 can be viewed as a reduction from the required deposit amount to certain per-deposit payment value rates: A (for the average case analysis), which is the maximum cumulative value of probabilistic payments across merchants within any detection time period; or W (for the worst case analysis), which is the maximum cumulative value of macropayments across merchants within the same period. Our analysis is *tight*, so leaving these parameters unbounded enables a malicious customer to gain unbounded additional utility (and rules out the possibility of deterring malicious behavior via economic means such as advance deposits). The purpose of this section is to discuss the meaning of bounding payment value rates, and what are the implications of such bounds. Throughout, recall that our analysis is *per deposit*, so we fix a single deposit $\mathbf{d}$ that backs all the probabilistic payments discussed below.

Interpretation of the parameters. We first discuss the detection time (used to define the rate), and then discuss how W and A may arise as a sum, across all merchants, of corresponding payment value rates.

- *Detection time.* We denote by T_{M} the *time for a merchant* M *to detect a detectable double spend*. For example, T_{M} can be the network's broadcast time, that is, the time for a message sent by a merchant to reach all other merchants (this is true, e.g., if the network contains enough honest nodes to provide reliable and timely broadcast, or if merchants have the same view of the ledger). In a Bitcoin-like system the broadcast time is much smaller than the validation time (the time for a broadcast transaction to appear in the ledger): a few seconds as opposed to a few minutes.
- *Merchants (per deposit).* We denote by N the *number of merchants* that accept probabilistic payments (backed by the deposit $\mathbf{d}$). For example, N could be the number of all merchants. (Though this need not be the case, see below.)
- *Payment value rates (per deposit).* For every merchant M, $\mathfrak{a}_{\mathsf{M}} := \max_t \mathfrak{a}_{\mathsf{M}}(t)$ is the maximum (cumulative) value of probabilistic payments (backed by the deposit $\mathbf{d}$) accepted by M within any time period of T_{M}; similarly, $\mathfrak{w}_{\mathsf{M}} := \max_t \mathfrak{w}_{\mathsf{M}}(t)$ is the maximum (cumulative) value of macropayments accepted by M within any time period of T_{M}. Then one sets A equal to $\sum_{\mathsf{M}} \mathfrak{a}_{\mathsf{M}}$, and W equal to $\sum_{\mathsf{M}} \mathfrak{w}_{\mathsf{M}}$ (or consider these as upper bounds to A and W).

Necessity of bounds. We now explain why simultaneous bounds on the aforementioned parameters are necessary. First, if there is no bound on the number N of merchants that accept probabilistic payments backed by $\mathbf{d}$, a malicious customer can use $\mathbf{d}$ to gain unbounded additional utility via a one-ticket-multiple-merchant attack (see Section 3.1), even in the average case. Second, even if N is bounded (and greater than 1) but $\max_{\mathsf{M}} T_{\mathsf{M}}$ is unbounded (e.g., a large-scale eclipse attack is underway [HKZG15]), a malicious

customer can gain unbounded additional utility via a multiple-ticket-multiple-merchant attack (see Section 3.1), even in the average case. Third, even if N and $\max_{\mathsf{M}} T_{\mathsf{M}}$ are bounded but some $\mathfrak{a}_{\mathsf{M}}$ is unbounded, our analysis implies that a malicious customer can again gain unbounded additional utility in the average case; similarly, if some $\mathfrak{w}_{\mathsf{M}}$ is unbounded, our analysis implies that a malicious customer can again gain unbounded additional utility in the worst case. In sum, if either $\max_{\mathsf{M}} T_{\mathsf{M}}$ or N are unbounded, then $A = \sum_{\mathsf{M}} \mathfrak{a}_{\mathsf{M}}$ and $W = \sum_{\mathsf{M}} \mathfrak{w}_{\mathsf{M}}$ are also unbounded; but even if $\max_{\mathsf{M}} T_{\mathsf{M}}$ and N are bounded, either A or W could still be unbounded, and so we must explicitly bound A or W (depending if we target average or worst case, or both).

Finally, observe that the above discussion assumes that there is no a-priori bound on how many tickets a single deposit can back; see Remark 2 below for a discussion of what happens if a deposit is restricted to only back macropayments up to a certain maximum total value.

Respecting the bounds. Whose responsibility is it to ensure that the bounds A or W are respected? One answer to this question could be that there are exogenous reasons (e.g., spending patterns, network behavior, and so on) that justify this statement. Another answer to this question is to say that every merchant M is responsible "for his own share": he needs to monitor that $\mathfrak{a}_{\mathsf{M}}$ and $\mathfrak{w}_{\mathsf{M}}$ are locally respected for him (and if they are about to be exceeded, he defers further payments to the next period of time T_{M}). This second answer raises an interesting technical problem: how does M know which payments are backed by the same deposit? If a payment's deposit is not private (as in [PS16]) this is not a problem. But if a payment's deposit is private, this could be tricky. In our DAM scheme construction, when engaging in a probabilistic payment, a merchant does not learn any information about the deposit that backs it, beyond the bit of whether the deposit is valid or not. Nevertheless, we still enable a merchant to get around this problem, by leveraging the notion of a *rate limit tag* within a probabilistic payment.

Implications: good news and bad news. The good news about our economic analysis is that it gives a tight characterization of the additional utility that can be gained via double spending. The bad news is that bounding A or W may impact usability. (Perhaps this is not surprising because offline probabilistic payments are a "tough" setting since double spending cannot be fully prevented.) Namely, if all $\mathfrak{a}_{\mathsf{M}}$ (resp., $\mathfrak{w}_{\mathsf{M}}$) are large, then A (resp., W) is even larger; but this impacts usability because the required deposit is large. Conversely, if many $\mathfrak{a}_{\mathsf{M}}$ (resp., $\mathfrak{w}_{\mathsf{M}}$) are small then A (resp., W), and thus the required deposit, is not as large; but the amount of value transacted with many merchants is limited, and this impacts usability because a user may not be able to transact large amounts with his "favorite" merchants.

Mitigations. A way to mitigate the above problem is to associate to each deposit a subset $\mathcal{R}$ of allowed "receiver merchants" so that the sum is taken only over this subset: $A = \sum_{\mathsf{M} \in \mathcal{R}} \mathfrak{a}_{\mathsf{M}}$ and $W = \sum_{\mathsf{M} \in \mathcal{R}} \mathfrak{w}_{\mathsf{M}}$. Then, any particular user would only have to cover his spending habits with one (or more) deposits that cover one (or more) not-too-large subsets of merchants. The subset $\mathcal{R}$ can even be private and chosen by the user; in fact, we take this approach both when defining and constructing a DAM scheme.

Another way to mitigate the above problem is for merchants to group together into *micropayment agencies*. Such an agency acts as a proxy to the subset of merchants it serves, and its only task is to "monitor" the cumulative values of $\mathfrak{a}_{\mathsf{M}}$ and $\mathfrak{w}_{\mathsf{M}}$ for

merchants in the agency. This approach does not affect any privacy guarantees from the perspective of the customer (since every probabilistic payment is anonymous from the perspective of a single merchant or any coalition of merchants). In the extreme, one could even think of a *single* micropayment agency, and the only obstacle would be coordinating and keeping track of A and W across the network.

Remark 2 (bounded macropayments per deposit). So far we have assumed that there is no a-priori bound on how many tickets a single deposit can back. Suppose instead that a deposit $\mathbf{d}$ can only back tickets with total macropayment value up to V_{tot}. To analyze this other setting, we can reuse ideas from our economic analysis: again, one can define additional utility by comparing the utilities of a malicious merchant and a corresponding honest merchant. We omit the analysis and simply state that the additional utility is bounded by $(2N-1)V_{\mathsf{tot}}$, where N is the number of merchants that accept probabilistic payments backed by $\mathbf{d}$ (note that in this case $\max_{\mathsf{M}} T_{\mathsf{M}}$, A, W could all be unbounded). Moreover, the bound is tight; intuitively, the maximum additional utility is achieved via a multiple-ticket-multiple-merchant attack until two macropayments with the same ticket occur for each of the N merchants.

4 Efficient fractional message transfer

A key ingredient in our construction of a DAM scheme is *fractional message transfer* (FMT): a primitive that enables probabilistic message transmission between two parties, called the 'sender' and the 'receiver'. Informally, the receiver samples a one-time key pair based on a transfer probability p; then, the sender uses the receiver's public key to encrypt a message m into a ciphertext c; finally, the receiver uses the secret key to decrypt c, thereby learning m, but only with the pre-defined probability p (and learns no information about m with probability $1-p$). Our definition and construction of FMT are closely related to *non-interactive fractional oblivious transfer* (NFOT), which was studied in the context of 'translucent cryptography' as an alternative to key escrow [BM89, BR99]; see Section 1.1.3 for a discussion.

In this work we formulate the notion of an *FMT scheme*, which formally captures the functionality and security of probabilistic message transmission; we rely on this tool (and others) in our construction of a DAM scheme. Moreover, we give an efficient construction of an FMT scheme that works for transfer probabilities $p = 1/n$ with $n \in \mathbb{N}$; this construction is in the random oracle model and assumes the hardness of the DDH problem in prime-order groups. Finally, since probabilistic message transmission is of independent interest, we also define the notion of an *FMT protocol* via an ideal functionality, and show that the security definition of FMT schemes does imply security relative to that ideal functionality. (Our DAM scheme relies on an FMT scheme, rather than an FMT protocol, because we interleave the FMT scheme with other building blocks.)

We defer the definitions, constructions, and proofs about FMT to the full version. In the rest of this section, we informally describe the syntax, correctness, and security of FMT schemes, and then sketch our FMT construction.

Syntax. An FMT scheme is a quintuple of algorithms $(\mathsf{FMT.Setup}, \mathsf{FMT.Keygen}, \mathsf{FMT.Encrypt}, \mathsf{FMT.Decrypt})$ with the following syntax.

- *Parameter setup (executed by a trusted party):* $\mathsf{FMT.Setup}(1^\lambda) \to \mathsf{pp}_{\mathsf{FMT}}$. On input a security parameter λ, $\mathsf{FMT.Setup}$ outputs the public parameters $\mathsf{pp}_{\mathsf{FMT}}$ for the scheme.
- *Key generation (executed by the receiver):* $\mathsf{FMT.Keygen}(\mathsf{pp}_{\mathsf{FMT}}, p) \to (\mathsf{pk}_{\mathsf{FMT}}, \mathsf{sk}_{\mathsf{FMT}})$. On input public parameters $\mathsf{pp}_{\mathsf{FMT}}$ and a transfer probability p, $\mathsf{FMT.Keygen}$ outputs a one-time key pair $(\mathsf{pk}_{\mathsf{FMT}}, \mathsf{sk}_{\mathsf{FMT}})$.
- *Message encryption (executed by the sender):* $\mathsf{FMT.Encrypt}(\mathsf{pp}_{\mathsf{FMT}}, \mathsf{pk}_{\mathsf{FMT}}, m) \to c$. On input public parameters $\mathsf{pp}_{\mathsf{FMT}}$, a public key $\mathsf{pk}_{\mathsf{FMT}}$ and a message m, $\mathsf{FMT.Encrypt}$ outputs a ciphertext c.
- *Message decryption (executed by the receiver):* $\mathsf{FMT.Decrypt}(\mathsf{pp}_{\mathsf{FMT}}, \mathsf{sk}_{\mathsf{FMT}}, c) \to m'$. On input public parameters $\mathsf{pp}_{\mathsf{FMT}}$, a secret key $\mathsf{sk}_{\mathsf{FMT}}$ and a ciphertext c, $\mathsf{FMT.Decrypt}$ outputs a message m' that equals m or $\varnothing$. (The special symbol $\varnothing$ denotes that decryption resulted in no message.)

An FMT scheme satisfies the correctness and security properties defined below.

Correctness. An FMT scheme is *correct* if for every security parameter λ, public parameters $\mathsf{pp}_{\mathsf{FMT}} \in \mathsf{FMT.Setup}(1^\lambda)$, transfer probability $p \in \mathcal{P} \subseteq [0,1]$, key pair $(\mathsf{pk}_{\mathsf{FMT}}, \mathsf{sk}_{\mathsf{FMT}}) \in \mathsf{FMT.Keygen}(\mathsf{pp}_{\mathsf{FMT}}, p)$, and message $m \in \mathcal{M}$,

$$\mathsf{FMT.Decrypt}(\mathsf{pp}_{\mathsf{FMT}}, \mathsf{sk}_{\mathsf{FMT}}, \mathsf{FMT.Encrypt}(\mathsf{pp}_{\mathsf{FMT}}, \mathsf{pk}_{\mathsf{FMT}}, m)) = \begin{cases} m & \text{w.p.} \quad p \\ \varnothing & \text{w.p.} \quad 1-p \end{cases}$$

where the probability is taken over the randomness of $\mathsf{FMT.Encrypt}$ (and $\mathsf{FMT.Decrypt}$ is deterministic).

Security. An FMT scheme is *secure* if it has the properties of *fractional hiding* and *fractional binding*. Informally, fractional hiding says that an honest encryptor transferring a message m can be sure that the decryptor, who knows the secret key, learns m with probability exactly p (and $\varnothing$ with probability $1-p$), even if the public key was generated maliciously. Fractional binding says that, for every $p' \neq p$, a malicious encryptor cannot produce a valid ciphertext that decrypts with probability p' to a valid message (i.e., not $\varnothing$).

An efficient FMT scheme. Our construction of an FMT scheme targets the case where p equals $1/n$ for some positive integer n; this case suffices within our construction of a DAP scheme. As in prior work [BM89, BR99], our starting point is the Elgamal encryption scheme [Elg85], whose semantic security relies on the hardness of DDH in prime-order groups. We now give an informal sketch of our construction.

- $\mathsf{FMT.Setup}(1^\lambda)$: sample a group $\mathbb{G}$ of prime order q (depending on λ), along with two generators $g, g_0 \in \mathbb{G}$.
- $\mathsf{FMT.Keygen}(\mathsf{pp}_{\mathsf{FMT}}, p)$: the public key contains a Pedersen commitment [Ped91] to a random s in $\{1, \ldots, n\}$ and the secret key contains the commitment's randomness; that is, the commitment is $h = g_0^{-s} g^\alpha$ for random $\alpha \in \mathbb{Z}_q$.
- $\mathsf{FMT.Encrypt}(\mathsf{pp}_{\mathsf{FMT}}, \mathsf{pk}_{\mathsf{FMT}}, m)$: sample random $r \in \mathbb{Z}_q$ and random $t \in \{1, \ldots, n\}$, and use h as an Elgamal public key to encrypt the message $m' := m \cdot g_0^{rt}$; the resulting ciphertext is $c = (t, c_1, c_2) = (t, g^r, m' h^r)$.

– $\mathsf{FMT.Decrypt}(\mathsf{pp}_{\mathsf{FMT}}, \mathsf{sk}_{\mathsf{FMT}}, c)$: use the secret key α to decrypt the ciphertext by setting $m'' := c_2/c_1^{\alpha} = mg_0^{r(t-s)}$.

The above sketch omits several important details. In particular, our construction also includes NIZKs (obtained via the Fiat–Shamir transform applied to simple Σ-protocols) to prove correctness of key generation and encryption. Informally, our FMT's correctness and security follow from the fact that $m'' = m$ only when $t = s$, which occurs with probability $p = 1/n$. The full construction and proof of security (based on hardness of DDH) are the full version.

5 Informal construction description

Recall that a DAM scheme is a tuple of algorithms:

$$\mathsf{DAM} = \begin{pmatrix} \mathsf{Setup} & \mathsf{MintCoin}^{\mathbf{L}} & \mathsf{PourCoinToCoin}^{\mathbf{L}} & & \\ \mathsf{CreateAddr} & \mathsf{MintDeposit}^{\mathbf{L}} & \mathsf{PourCoinToDeposit}^{\mathbf{L}} & \mathsf{WithdrawDeposit}^{\mathbf{L}} & \mathsf{Punish} \\ \mathsf{Receive}^{\mathbf{L}} & \mathsf{MintTicket}^{\mathbf{L}} & \mathsf{PourCoinToTicket}^{\mathbf{L}} & \mathsf{RefreshTicket}^{\mathbf{L}} & \mathsf{VerifyTransaction}^{\mathbf{L}} \\ & & \mathsf{PourTicket}^{\mathbf{L}} & & \end{pmatrix}.$$

We sketch the construction of these in Section 5.1 and Section 5.2, and then separately discuss security intuition in Section 5.3 and 'pour regulation' in Section 5.4.

5.1 Informal algorithm descriptions

Setup. The algorithm DAM.Setup samples public parameters for the various building blocks that we use, which includes DAP schemes and FMT schemes, as well as one-time signature schemes and NIZKs

Creating addresses. The algorithm DAM.CreateAddr samples a new address key pair by running DAP.CreateAddr with the address information set to the probabilistic payment specification and outputting its result; in other words, DAM addresses are simply addresses of the underlying DAP scheme. Receivers must bind their intended payment rates, probability, and value to the address by passing it as input to DAM.CreateAddr (other users can set it to $\perp$ if they do not intend to receive probabilistic payments).

Receiving coins. The algorithm DAM.Receive, given an address key pair, retrieves all the unspent coins sent to this address by simply running DAP.Receive. Indeed, DAM pour-coin, pour-ticket, withdraw, and refresh transactions can be viewed as DAP pour transactions, and so DAP.Receive may retrieve from these any relevant coins.

Minting notes. Each of the minting algorithms DAM.MintCoin, DAM.MintDeposit, and DAM.MintTicket first sets the public information string pub to the type of the note being minted (respectively, `cn`, `dp`, or `tk`), and sets the secret information string sec accordingly: for coins, sec equals $\perp$; for deposits, sec is a commitment to the deposit's receiver address set $\mathcal{R}$; for tickets, sec equals the (already-minted) deposit that backs it. Then, the algorithm mints the note by running DAP.Mint.

Pouring coins. Each of the algorithms DAM.PourCoinToCoin, DAM.PourCoinToDeposit, DAM.PourCoinToTicket first sets the public and secret information strings similarly to above, and then runs DAP.Pour to generate the new notes. A DAM scheme also includes a protocol for pouring tickets into coins, which we discuss separately in Section 5.2 because it is the most complex part of the construction.

Withdrawing deposits. The algorithm DAM.WithdrawDeposit, given a deposit $\mathbf{d}$ (and its address secret key) and address public key apk, pours the deposit into a new coin $\mathbf{c}$ with address apk by running DAP.Pour. The output consists of the new coin $\mathbf{c}$, as well as a withdraw transaction $\mathsf{tx}_{\mathsf{wd}}$ that is just a DAP pour transaction having activation delay Δ_{w}. Since pour transactions reveal the serial numbers of input notes, it is easy to blacklist the withdrawn deposit (see Section 5.2).

5.2 A 3-message protocol for probabilistic payments

We outline the construction of DAM.PourTicket, a 3-message protocol that realizes an offline probabilistic payment between a sender (customer) and receiver (merchant). For simplicity, we only discuss enforcement of the worst-case payment rate bounds; enforcement of the average-case bound is achieved via essentially the same ideas. Recall that the worst-case bound limits the number of macropayments that occur in a particular time window tw.

1st message (sender ← receiver). The first message of the protocol is from the receiver to the sender and consists of the receiver's *session identifier* sid, *session public key* spk, the list of deposits $\mathcal{D}$ that have been blacklisted in this epoch, and the desired public value v_{pub}. These are constructed as follows. Suppose that the receiver has an address key pair $(\mathsf{apk}_{\mathbf{c}}, \mathsf{ask}_{\mathbf{c}})$ and wishes to receive payments at this address with payment probability p_{r} and macropayment value V_{r}; moreover, suppose that the receiver's per-deposit maximum cumulative average-case payment value rate is $\mathfrak{a}_{\mathsf{r}}$. Then the receiver constructs his session identifier as $\mathsf{sid} := (\mathsf{apk}_{\mathbf{c}}, \mathsf{tw}_{\mathsf{r}})$. To construct the session public key, the receiver samples a new key pair $(\mathsf{pk}_{\mathsf{SIG}}, \mathsf{sk}_{\mathsf{SIG}})$ for the one-time signature scheme, and a new key pair $(\mathsf{pk}_{\mathsf{FMT}}, \mathsf{sk}_{\mathsf{FMT}})$ for the fractional message transfer scheme and sets $\mathsf{spk} := (\mathsf{pk}_{\mathsf{FMT}}, \mathsf{pk}_{\mathsf{SIG}})$. Finally, the deposit blacklist $\mathcal{D}$ consists of the identifiers of deposits seen in punish transactions within the *current* epoch.

2nd message (sender → receiver). The sender now pours his ticket $\mathbf{t}$ into a new coin $\mathbf{c}$ using DAP.Pour, and then uses fractional message transfer to probabilistically transmit the new coin $\mathbf{c}$ to the receiver, while also proving, in zero knowledge, that he did so correctly. We now expand on this description, which hides subtle aspects of our construction.

After pouring his ticket $\mathbf{t}$ into a new coin $\mathbf{c}$ (which results in a DAP pour transaction tx_{p}), the sender uses the deposit $\mathbf{d}$ backing $\mathbf{t}$ to generate two crucial quantities: the worst-case *rate limit tag* wrlt and the *double spend tag* dst. The rate limit tag allows the receiver to enforce the payment value rate bounds required by the economic analysis. The double spend tag allows the receiver to extract deposit revocation information if and only if $\mathbf{t}$ is spent in two macropayments.

A natural strategy would be for the sender to send to the receiver, in the clear, the rate limit tag wrlt, and a FMT ciphertext c_{FMT} containing tx_{p} and dst, along with a non-interactive zero knowledge proof that both were generated correctly. However, doing so does not preserve privacy. Indeed, to ensure that the sender cannot double spend the ticket to herself and escape punishment, the ledger needs to check that the double spend tag was generated correctly. This can be done by verifying the NIZK proof, but to do this would require including the FMT ciphertext, blacklist detection tag, and rate limit tag as part of the NP instance being verified. This is problematic, since *publishing these*

leaks information about the transfer probability p_r and the deposit, both of which are private information.

To fix this problem, the sender hides wrlt and c_{FMT} inside two commitments ω_0 and ω_1, and then computes a proof of correctness relative to these commitments. More precisely, the first commitment ω_0 hides $m_0 := (\mathsf{sid}, \mathsf{spk}, v_{\mathsf{pub}}, c_{\mathsf{FMT}})$, where sid, spk, and v_{pub} are the receiver's session identifier, session public key, and public value respectively, and c_{FMT} is a FMT ciphertext. The FMT ciphertext c_{FMT}, as before, contains wrlt, $\mathsf{tx_p}$ and dst, but now also contains randomness r_1 that opens the second commitment ω_1, which in turn hides $m_1 := (\mathsf{tx_p}, \mathsf{dst}, \mathsf{wrlt})$. Thus opening the FMT ciphertext allows the receiver to open ω_1 and obtain the correct $\mathsf{tx_p}, \mathsf{dst}, \mathsf{wrlt}$. Next, the sender generates a non-interactive zero knowledge proof of knowledge π_{pt} asserting that he performed all these steps correctly. The NIZK also asserts (a) that the deposit $\mathbf{d}$'s receiver address set $\mathcal{R}$ contains the receiver's address public key apk, (b) that $\mathbf{d}$'s identifier has not appeared in punish transactions in the current epoch, and (c) that $\mathbf{d}$'s serial number has not appeared on the ledger prior to the current epoch (that is, $\mathbf{d}$ was not revoked or withdrawn in prior epochs).

Finally, he sends $(\omega_0, m_0, \omega_1, \pi_{\mathsf{pt}})$ and randomness r_0 for opening ω_0 to the receiver. Since the proof is now computed relative to ω_0 and ω_1, and not c_{FMT} and wrlt, it can safely be published to the ledger.

3rd message (sender $\leftarrow$ receiver). The receiver uses r_0 and m_0 to open ω_0 and checks that the committed sid, spk and v_{pub} are indeed the correct ones (which were sent in the first message). Next, he checks the correctness of π_{pt}, and finally, using the rate limit tag, he checks that the payment value rate $\mathfrak{a}_r$ has not been exceeded. If these checks pass, he tries to open the FMT ciphertext c_{FMT} inside ω_0. If he is able to successfully open it, he can open ω_1 to obtain $\mathsf{tx_p}$ and dst. If the ticket $\mathbf{t}$ has already been spent (i.e., the deposit $\mathbf{d}$ has been blacklisted), the receiver recovers the deposit and creates a punish transaction $\mathsf{tx_{pun}}$. If not, he posts $\mathsf{tx_p}$ to obtain his payment. Finally, he sends to the sender the secret key $\mathsf{sk_{FMT}}$ used for decryption, and m', which is the outcome of decryption, to communicate whether the outcome was 'macropayment' or 'nullpayment'.

Outcome verification. Upon receiving the FMT secret key, the sender checks that the FMT ciphertext c_{FMT} decrypts to claimed message m' under the key $\mathsf{sk_{FMT}}$; this reveals whether the claimed outcome was the correct one. If the receiver sends an incorrect secret key, or does not send anything at all, the sender refreshes his ticket, thereby generating a new ticket $\mathbf{t}'$ and a refresh transaction $\mathsf{tx_{ref}}$.

5.3 Security considerations

We give an intuitive justification of why the probabilistic payment protocol is secure.

Sender security. The fractional hiding property of the FMT scheme ensures that the receiver can only open c_{FMT} with probability p_r. Since the commitment ω_1 is hiding and the proof π_{pt} is zero knowledge, the rest of the sender's message is indistinguishable from random. Finally, the security of the "outcome verification" step is guaranteed by the fractional hiding property of the FMT scheme; if the receiver could generate two different secret keys that can decrypt the same FMT ciphertext to different messages, then he could bias the probability of opening the ciphertext in his favor, thus breaking fractional hiding.

The above ensures "intra-protocol" sender security. Post-protocol security requires that the receiver cannot compromise the honest sender's anonymity or cause monetary loss by aborting. This is achieved by allowing the sender to refresh tickets by pouring them into new ones. This breaks the link between the ticket that the receiver has seen and the ticket that the sender can now spend, enabling the sender to freely spend his new ticket.

Receiver security. Opening ω_0 allows the receiver to check that the sender generated the rate limit tags relative to the true session identifier and public key. The fractional binding property of the FMT scheme ensures that the sender cannot alter the probability of opening c_{FMT}. The NIZK proof ensures the correctness of each step.

The above ensures "intra-protocol" receiver security. Achieving post-protocol security is trickier, since we need to ensure that the sender can only create double spend tags that are consistent across independent pour-ticket transactions. In our construction, the sender can attempt to bypass this requirement by manipulating the three inputs that create a double spend tag: the randomness x used for generating the tag, and the deposit $\mathbf{d}$ that is hidden in the tag, and the ticket $\mathbf{t}$ that $\mathbf{d}$ backs.

Preventing reuse of randomness. To prevent recovery of the deposit serial number from multiple double spend tags, the sender could attempt to reuse randomness across each tag. This would prevent recovery, since each receiver would possess the same tag. To prevent this, our construction of a double spend tag dst uses a special one-time signature public key $\mathsf{pk}_{\mathsf{SIG}}$ as randomness. Later, upon receiving the tag, the receiver signs the tag (among other things) with the secret key $\mathsf{sk}_{\mathsf{SIG}}$ corresponding to $\mathsf{pk}_{\mathsf{SIG}}$. To create two different pour-ticket transactions with the same double spend tag (one to an honest receiver and one back to himself), the sender would thus have to forge a signature, which is computationally infeasible by the security of the signature scheme.
Ensuring $\mathbf{d}$ *backs* $\mathbf{t}$. The NIZK proof created by the sender ensures that the deposit $\mathbf{d}$ hidden in the double spend tag is the one backing $\mathbf{t}$.
'Identical' tickets backed by different deposits. In principle, one could construct two tickets $\mathbf{t}, \mathbf{t}'$ that have the same serial number (and are thus indistinguishable from the point of view of double spending), but are backed by different deposits. Since $\mathbf{t}$ and $\mathbf{t}'$ would share serial numbers, only one of the two could be successfully spent. This could lead to the following attack: the sender generates two such tickets, and pays himself with one, and pays a receiver with the other. When a macropayment occurs, he front runs the receiver to get his self-payment onto the ledger first. The receiver is then robbed of his payment, but also cannot punish the sender, since the double spend tags hide different deposits, making revocation impossible.
However, our construction prevents such an attack by ensuring that the serial number of a note is derived (in part) from its secret information string sec. This property is guaranteed by the DAP scheme.

5.4 Regulating type transitions when pouring

The definition of a DAM scheme restricts fund transfers between different note types: coins can be poured into coins, deposits, or tickets; a deposit can be poured into a coin; a ticket can be poured into a coin or ticket. Moreover, some type transitions are handled differently from others: for example, pouring from a set of coins yields a pour-coin

transaction that is immediately valid, while pouring from a ticket to a ticket yields a refresh transaction that only becomes valid after a waiting period (the activation delay). We realize most of these fund transfers via DAP pours, but we must also somehow meet the aforementioned restrictions.

The first obstacle is that a note's type is not necessarily known, because we store the type of note in its public information string pub, which is not revealed by a DAP pour transaction. But remember that a DAP scheme allows us to choose, at parameter setup time, a pour predicate that regulates all pour transactions. We thus engineer a pour predicate Π^*_{p}, tailored for our application, that (i) allows only the aforementioned type transitions, and (ii) ensures that the information string info in a DAP pour transaction correctly exposes the type of the note from which we are pouring.

References

BBSU12. Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better - how to make Bitcoin a better currency. In *FC '12*, 2012.

BCG+14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *SP '14*, 2014.

Bit13. Bitcoinj. Working with micropayment channels. `https://bitcoinj.github.io/working-with-micropayments`, 2013.

Blo14. Block Chain Analysis. Block chain analysis. `http://www.block-chain-analysis.com/`, 2014.

BM89. Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. CRYPTO '89, 1989.

BP15. Alex Biryukov and Ivan Pustogarov. Proof-of-work as anonymous micropayment: Rewarding a Tor relay. In *FC '15*, 2015.

BR99. Mihir Bellare and Ronald L. Rivest. Translucent cryptography - an alternative to key escrow, and its implementation via fractional oblivious transfer. *Journal of Cryptology*, 1999.

Cal12. Mike Caldwell. Sustainable nanopayment idea: Probabilistic payments. `https://bitcointalk.org/index.php?topic=62558.0`, 2012.

Cha82. David Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, 1982.

Cha15. Chainalysis. Chainalysis inc. `https://chainalysis.com/`, 2015.

CHL05. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *EUROCRYPT '05*, 2005.

DFKP13. George Danezis, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio Coin: building Zerocoin from a succinct pairing-based proof system. In *PETShop '13*, 2013.

DW15. Christian Decker and Roger Wattenhofer. A fast and scalable payment network with Bitcoin duplex micropayment channels. In *SSS '15*, 2015.

Elg85. Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 1985.

Ell13. Elliptic. Elliptic enterprises limited. `https://www.elliptic.co/`, 2013.

GM16. Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. ePrint 2016/701, 2016.

HAB+16. Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted Bitcoin-compatible anonymous payment hub. ePrint 2016/575, 2016.

HKZG15. Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *Security '15*, 2015.

HS12. Mike Hearn and Jeremy Spilman. Bitcoin contracts. `https://en.bitcoin.it/wiki/Contract`, 2012.

KMS+16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *SP '16*, 2016.

LO98. Richard J. Lipton and Rafail Ostrovsky. Micropayments via efficient coin-flipping. In *FC '98*, 1998.

MB15. Malte Möser and Rainer Böhme. Trends, tips, tolls: A longitudinal study of Bitcoin transaction fees. In *Bitcoin '15*, 2015.

MGGR13. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from Bitcoin. In *SP '13*, 2013.

Mic14. Silvio Micali. Universal payment systems. `https://www.youtube.com/watch?v=xgA6TO7drok`, 2014.

MPJ+13. Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of Bitcoins: characterizing payments among men with no names. In *IMC '13*, 2013.

MR02. Silvio Micali and Ronald L. Rivest. Micropayments revisited. In *CT-RSA '02*, 2002.

MRK03. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *FOCS '03*, 2003.

Nak09. Satoshi Nakamoto. Bitcoin: a peer-to-peer electronic cash system, 2009. URL: `http://www.bitcoin.org/bitcoin.pdf`.

PD16. Joseph Poon and Thaddeus Dryja. The Bitcoin lightning network: Scalable off-chain instant payments. `https://lightning.network/lightning-network-paper.pdf`, 2016.

Ped91. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, 1991.

PS15. Rafael Pass and Abhi Shelat. Micropayments for decentralized currencies. In *CCS '15*, 2015.

PS16. Rafael Pass and Abhi Shelat. Micropayments for decentralized currencies. ePrint 2016/332, 2016.

RH11. Fergal Reid and Martin Harrigan. An analysis of anonymity in the Bitcoin system. In *SocialCom/PASSAT '11*, 2011.

Riv97. Ronald L. Rivest. Electronic lottery tickets as micropayments. In *FC '97*, 1997.

Riv04. Ronald L. Rivest. Peppercoin micropayments. In *FC '04*, 2004.

RKS15. Tim Ruffing, Aniket Kate, and Dominique Schröder. Liar, liar, coins on fire!: Penalizing equivocation by loss of Bitcoins. In *CCS '15*, 2015.

RS13. Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. In *FC '13*, 2013.

ST99. Tomas Sander and Amnon Ta-Shma. Auditable, anonymous electronic cash extended abstract. In *CRYPTO '99*, 1999.

vOR+03. Nicko van Someren, Andrew M. Odlyzko, Ronald L. Rivest, Tim Jones, and Duncan Goldie-Scot. Does anyone really need micropayments? In *FC '03*, 2003.

Whe96. David Wheeler. Transactions using bets. In *SPW '96*, 1996.

Yao77. Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS '77*, 1977.