Spear: Fast Multi-Path Payment with Redundancy

Sonbol Rahimpour
University of Alberta
rahimpou@ualberta.ca

Majid Khabbazian
University of Alberta
mkhabbazian@ualberta.ca

ABSTRACT

In a payment network, like the Lightning Network, Alice can transfer a payment to Bob by splitting the payment into partial payments and transferring these partial payments through multiple paths. The transfer, however, delays if any of the partial payments fails or delays. To handle this, one can add redundant payment paths. The challenge in doing so is that Bob may now overdraw funds from the redundant paths. To address this, Bagaria, Neu, and Tse introduced Boomerang, a mechanism based on secret sharing and homomorphic one-way functions, which allows Alice to revert the transfer if Bob overdraws.

In this work, we introduce Spear, a simple method with lower latency than Boomerang. In addition, Spear needs significantly less computation, and half the maximum locktime of Boomerang. Unlike Boomerang, Spear can be implemented using only a minor change to the Lightning Network. This minor change enables both Alice and Bob to have control over the release of partial payments. This prevents Bob from ever overdrawing. Another interesting feature of Spear is that it is more robust than Boomerang against malicious intermediate nodes who do not forward payments in an attempt to lock up funds. Finally, Spear trivially supports division of a payment into *uneven* partial payments. This gives Alice maximum flexibility in dividing her payment into partial payments.

KEYWORDS

Lightning Network, routing, multi-path payment, redundancy

ACM Reference Format:
Sonbol Rahimpour and Majid Khabbazian. 2021. Spear: Fast Multi-Path Payment with Redundancy. In *Proceedings of ACM AFT (AFT'20).* ACM, New York, NY, USA, 9 pages. https://doi.org/nn.nnnn/nnnnnnn.nnnnnnn

1 INTRODUCTION

Permissionless blockchains can provide trust in trustless environment. As a result, many applications such as financial applications can rely on a blockchain rather than trusted third parties such as banks. This shift in paradigm has brought a lot of interest and attention to blockchain. Blockchain applications such as Bitcoin are not, however, quite ready for mainstream use because they have scaling issues. Bitcoin, for example, can handle up to ten transactions per second, and needs, on average, at least ten minutes to add a transaction. Custodian payment systems such as Visa, on the other hand, can handle tens of thousands of transactions per second all with a short confirmation time.

Blockchain's moon race for a scalable solution has led to a rich body of litrature, with solutions and proposals ranging from improving the consensus algorithms, to sharding [11, 13] and sidechains [2]. Among these solutions, "layer-two protocols" are among the most promising ones. These protocols allow users to conduct so-called *off-chain* transactions among themselves, without requiring to add these transactions to the blockchain.

There are several off-chain protocols in the literature, including payments [4, 8, 20, 25], state [16], and commit-chain [10, 24]. In this work, we focus on *payment channel network* (PCN) [25], and, in particular, the Lightning Network [12], which is the most prominent PCN for Bitcoin. In the Lightning Network, a user, say Alice, can join the network by connecting to an existing user, say Carol, through a so-called payment channel. Alice opens this payment channel by essentially locking up a fund, which is controlled by both parties. As soon as the channel is opened between Alice and Carol, the two parties can transfer funds between themselves by simply readjusting the fund allocation on the channel.

In the Lightning Network, Alice can use her channel with Carol to make a payment to another party, say Bob, who is already connected to the network. To perform this, Alice first selects a path of payment channels P from herself to Bob, and then transfers the payment to Bob through P. Because of privacy concerns, the balance of channels in the Lightning Network are considered private information. Since Alice is not aware of balances of channels on P (except the first channel on P which belongs to her), her payment may fail because a channel on the path may not have enough balance to forward the payment. If Alice's payment fails, she should retry the transfer by choosing a different path. This adds a non-negligible delay to the payment process, and increases the payment's time-to-complete.

To reduce the chance of payment failure (hence, reducing the time-to-complete), Alice may decide to split the payment into partial payments and transfer these partial payments through multiple paths. On the one hand, this helps as the smaller a payment, the more likely it is for the payment to go through. On the other hand, the whole payment will fail if any of the partial payments fails.

Similar conditions have emerged in other systems. For example, in storage systems, to improve performance, storage technologies such as RAID (Redundant Arrays of Independent Disks) use data striping, where data (e.g., a file) is distributed across several disks. This is analogous to distributing a payment into several partial payments. As in multi-path payment methods, where a single partial payment failure would fail the whole payment, in data storage systems, a single disk failure would result in data loss. To address this and improve dependability, RAIDs use redundant disks (as the name suggests). Interestingly, although RAID was invented to

AFT'20, October 2020, New York City

ACM ISBN xxx-x-xxxx-xxxx-x/YY/MM…$XX.00
https://doi.org/nn.nnnn/nnnnnnn.nnnnnnn

improve performance, its dependability, which is achieved through redundancy, is the key reason behind its widespread popularity [22].

Redundancy can bring advantages to multi-path payments, too. In [3], Bagaria, Neu, and Tse introduced Boomerang, the first multi-path payment method with redundant payments, and showed that it can improve throughput and time-to-completion of payments. The main challenge in supporting redundancy in a multi-path payment method is to protect Alice from overdrawing. To address this, Boomerang uses secret sharing to enable Alice to recover the payment when Bob overdraws. Interestingly, secret sharing is closely related to Reed-Solomon codes [15], an important group of erasure codes that enable data recovery in storage systems.

In this work, we take a simpler approach to add redundancy and protect Alice against overdrawing. While Boomerang's approach resembles coding techniques in data storage systems and communications, our solution (Spear) resembles ARQ (Automatic Repeat Request), a method that uses acknowledgements to achieve reliable data transmission over an unreliable communication channel. In ARQ, like Spear, the receiver informs the sender of the packets (the partial payments in the case of Spear) that it has received. Note that this method needs a duplex channel to allow the receiver to communicate with the sender. Such duplex channel does not always exists. For example, a data storage system is a one-way channel that communicates data from one point in time to another point in the future. In this case, a reverse channel does not exist, hence ARQ is not applicable. In payment networks, however, there is often a (out-of-band) channel between Alice and Bob through which they can exchange information such as hash values and, in the case of Boomerang, agree on the number of partial payments. In Spear, we use this out-of-band channel in a way similar to ARQ to achieve a higher performance than Boomerang.

Contributions. This paper makes the following contributions: We evaluate and compare the success probabilities of multi-path payment with and without redundancy, and demonstrate the superiority of the former. We introduce Spear, a simple multi-path payment method with redundancy. We prove Spear's various security guarantees, and explain how to implement Spear. Moreover, we compare Spear with its counterpart, Boomerang, against several measures including delay (i.e., time-to-complete), computational complexity, ease of implementation, maximum required locktime, resilience against intermediate node's misbehaviour, and flexibly. Our comparison results show that Spear improves Boomerang in all these measures, hence is a better choice in applications where there is an out-of-band channel between the payer and payee.

2 SYSTEM MODEL

We model the Lighting Network as a graph $G = (V, E)$, where V is the set of nodes and E is the set of channels. For any channel $(u, v) \in E$, let $b(u, v)$ denote the balance of node u, and $f(u, v, m)$ denote the fee that node u charges to forward a payment of amount m to v on channel (u, v). For each channel $(u, v) \in E$, the channel capacity is defined as $c(u, v) = b(u, v) + b(v, u)$.

Single-path payment. Let $\mathcal{P} = (v_1, v_2, \dots v_l, v_{l+1})$ be a path in G. A single-path payment of m from v_1 to v_{l+1} on path $\mathcal{P}$ is called *successful* if and only if

$$\forall 1 \le i \le l: \quad b(v_i, v_{i+1}) \ge m_i, \tag{1}$$

where $m_l = m$, and m_i, $i < l$, can be calculated recursively as

$$\forall 1 \le i < l: \quad m_i = m_{i+1} + f(v_i, v_{i+1}, m_{i+1}).$$

A successful transfer of payment m over $\mathcal{P}$ will result in the following changes in balances

$$\begin{aligned} \forall 1 \le i \le l: \quad b(v_i, v_{i+1}) &\longleftarrow b(v_i, v_{i+1}) - m_i \\ b(v_{i+1}, v_i) &\longleftarrow b(v_{i+1}, v_i) + m_i \end{aligned} \tag{2}$$

Multi-path payment. Another option for v_1 to transfer m to v_{l+1} is to split m into partial payments $m_1, m_2, \dots, m_k$, where $\sum_{i=1}^{k} m_i = m$, and transfer m_i, $1 \le i \le k$, on a path $\mathcal{P}_i$. Such multi-path payment is called *successful* if and only if sequentially transferring of m_i on $\mathcal{P}_i$ result in k successful transfers according to (1). Note that channel balances are updated according to (2) after a successful transfer of a partial payment and prior to the transfer of the next partial payment.

3 POWER OF REDUNDANCY

In [3], the authors experimentally showed that Boomerang, a multi-path payment with redundancy, can lead to 40% reduction in latency and 200% increase in throughput. In this section, we look at redundancy from a slightly different angle, and show its power in improving the success probability of payments.

We consider three payment options: 1) single-path payment; 2) multi-path payment without redundancy; 3) multi-path payment with redundancy. To show the power of redundancy, we compare the probability of success of these three options using a simple probabilistic model with the following assumptions:

(1) All channels have the same capacity C;
(2) There is no service fee;
(3) The balance of a channel is drawn uniformly at random independent of other channels.

We remark that this model serves to build intuition rather than to establish any fundamental impossibility results. In this model, we show that if multi-path payment without redundancy has a high probability of success so does single-path payment. In other words, multi-path payment without redundancy (in a high payment success probability regime) does not have a real advantage over single-path payment. We then show that if redundant payments are added, the success probability of multi-path payment can be significantly higher than that of a single-path payment. In particular, we show that there are scenarios in which multi-path payment with redundancy can achieve a high success rate while neither multi-path payment without redundancy nor single-path payment can. To get these results, we start with the following proposition.

Proposition 3.1. *Let $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_k$ be k paths from node A to node B. Let l_i denote the length of path $\mathcal{P}_i$ (i.e., l_i is the number of channels on $\mathcal{P}i$). Let P_k denote the probability that a payment of $m = \sum_{i=1}^{k} m_i$ can be transferred from A to B by simultaneously transferring partial payments of m_i on path $\mathcal{P}_i$.*
Then, we have

$$P_k = \Pi_{i=1}^{k} \left(1 - \frac{m_i}{C}\right)^{l_i},$$

if paths $\mathcal{P}_i$ are disjoint, and

$$P_k < \Pi_{i=1}^{k} \left(1 - \frac{m_i}{C}\right)^{l_i},$$

otherwise.

Proof. Suppose paths $\mathcal{P}_i$, $1 \le i \le k$, are disjoint. Let $\mathcal{E}_i$ be the event that the transfer of partial payment m_i on $\mathcal{P}_i$ is successful. We have

$$Pr(\mathcal{E}_i) = \left(1 - \frac{m_i}{C}\right)^{l_i},$$

where $\left(1 - \frac{m_i}{C}\right)$ is the probability that a channel has enough balance to forward m_i. The events $\mathcal{E}_i$, $1 \le i \le k$, are independent, because paths $\mathcal{P}_i$, $1 \le i \le k$, are disjoint. Therefore, we get

$$P_K = \Pi_{i=1}^{k} Pr\left(\mathcal{E}_i\right) = \Pi_{i=1}^{k} \left(1 - \frac{m_i}{C}\right)^{l_i}.$$

Now, let us consider the case where paths $\mathcal{P}_i$ are not disjoint. Consider any channel. The probability that this channel can transfer two partial payments m_i and m_j is

$$\left(1 - \frac{m_i + m_j}{C}\right)$$

which is less than

$$\left(1 - \frac{m_i}{C}\right) \cdot \left(1 - \frac{m_j}{C}\right).$$

This implies that transferring m_i and m_j on two different channels is more likely to succeed than transferring them on a single channel. Consequently, the success probability of transfer of payments is maximized when paths $\mathcal{P}_i$ are disjoint. □

Corollary 3.1.1. *Let l denote the length of the shortest path between A and B. Then, the probability that a payment m goes through a shortest path is*

$$P_{success}^{single} = \left(1 - \frac{m}{C}\right)^{l}.$$

Lemma 3.2. *Let l denote the length of the shortest path between A and B. Then we have*

$$P_{success}^{mult} < e^{-\frac{ml}{C}},$$

where $P_{success}^{mult}$ denotes the success probability of multi-path payment.

Proof. For any real number $0 < \alpha < 1$, and any integer $n \ge 1$, we have

$$(1 - \alpha)^n < e^{-\alpha n}.$$

Therefore, by Proposition 3.1, we get

$$\begin{aligned} P_{success}^{mult} &= \max_{k,m_i,l_i} \left\{ \Pi_{i=1}^{k} \left(1 - \frac{m_i}{C}\right)^{l_i} \right\} \\ &< \max_{k,m_i,l_i} \left\{ \Pi_{i=1}^{k} e^{-\frac{m_i l_i}{C}} \right\} \\ &\le \max_{k,m_i} \left\{ \Pi_{i=1}^{k} e^{-\frac{m_i l}{C}} \right\} \\ &= \max_{k,m_i} \left\{ e^{-\frac{\left(\sum_{i=1}^{k} m_i\right) l}{C}} \right\} \\ &= e^{-\frac{ml}{C}} \end{aligned}$$

□

Proposition 3.3. *We have*

$$P_{success}^{single} > 1 + \ln\left(P_{success}^{mult}\right)$$

Proof. By Corollary 3.1.1, the success probability of single-path payment is $\left(1 - \frac{m}{C}\right)^{l}$. Therefore

$$\begin{aligned} P_{success}^{single} &= \left(1 - \frac{m}{C}\right)^{l} \\ &\ge 1 - \frac{ml}{C} \\ &> 1 + \ln\left(P_{success}^{mult}\right), \end{aligned}$$

where the second inequality is by Lemma 3.2, and the first inequality is by the fact that $(1 - \alpha)^n \ge 1 - \alpha n$ for any real number $0 < \alpha < 1$, and any integer $n \ge 1$. □

Example 1. An important consequence of Proposition 3.3 is that if multi-path payment has a high probability of success so does single-path payment. For example, if $P_{success}^{mult} = 99\%$, then by Proposition 3.3, we get that $P_{success}^{single} > 98.99\%$. As another example, if $P_{success}^{mult} = 90\%$, then by Proposition 3.3, we get that $P_{success}^{single} > 89.46\%$. Note that this result holds for any values of m, l, and C.

Proposition 3.3 essentially shows that if single-path payment has low probability of success, we cannot expect multi-path payment to achieve a high probability of success. By adding redundant payments to our multi-path payment, however, we can achieve a high probability of success even when the success probability of single-path payment is low. Let us clarify this in the following example.

Example 2. Suppose $l = 4$, $\frac{m}{C} = \frac{1}{3}$, and there are $k = 10$ disjoint paths. Then, by Proposition 3.1, we get

$$P_{success}^{single} = \left(1 - \frac{m}{C}\right)^{l} = \left(1 - \frac{1}{3}\right)^{4} \approx 20\%$$

and

$$P_{success}^{mult} = \left(1 - \frac{m}{kC}\right)^{kl} = \left(1 - \frac{1}{10 \times 3}\right)^{10 \times 4} \approx 26\%$$

Now, if we use a multi-path payment method with redundant payments by transferring $\frac{m}{5}$ on every path, then the probability that at least five of these partial payments are successfully received can be shown to be at least 98%. This means that the probability of success of multi-path payment with redundancy can be higher than 98%, that is

$$P_{success}^{redundant} > 98\%,$$

where $P_{success}^{redundant}$ denotes the success probability of multi-payment with redundancy. Note that by Lemma 3.2, the probability of success of multi-path payment (without redundancy) for this example is at most

$$P_{success}^{mult} < e^{-\frac{ml}{C}} = e^{-\frac{4}{3}} \approx 26.36\%,$$

even when there are infinitely many disjoint paths.

4 SPEAR

An overview. Suppose Alice wishes to send a payment to Bob over a single path. To do so, Alice first acquires a hash digest from Bob. Then, she selects a single path to Bob, and sends the payment (conditioned on Bob disclosing the preimage) through the selected path. Finally, Bob accepts the payment by releasing his preimage. In

this single-payment scheme, Bob's hash digest can be used as part of an invoice, and its preimage can be treated as a receipt/proof of payment.

Spear can be viewed as an extension of the above single-payment scheme. It allows Alice to select multiple paths, including some redundant ones, to send partial payments to Bob. Spear follows the same procedure as the single-payment scheme, except it uses a slightly modified version of Hashed Timelock Contract (HTLC), which is the contract that conditions the release of payment on the disclosure of Bob's preimage. As will be explained later, the main purpose of using the new HTLC is to prevent Bob from overdrawing. We refer to this new HTLC as H^{2}TLC.

Figure 1 compares HTLC and H^{2}TLC. As reflected in the figure, the only difference between HTLC and H^{2}TLC is that H^{2}TLC uses two hash digests instead of one. In other words, the transfer of money in H^{2}TLC is conditioned on releasing two preimages instead of one. In Spear, one hash digest is set by Bob, while the second one is set by Alice. This simple addition gives both parties control over the release of the payment in the H^{2}TLC contract.

As in the single-payment scheme, in Spear Alice first acquires a hash digest from Bob. She then uses this hash digest in setting up H^{2}TLC in every selected path. The second hash digest in a H^{2}TLC is, however, set by Alice alone. Unlike Bob's hash digest which is the same on every path, Alice's hash digest varies from one path to another. When Bob receives multiple partial payments whose sum is equal to the whole payment, he would contact Alice over an out-of-band channel and ask her to provide him with the corresponding preimages. Alice will check whether the sum of the partial payments is indeed equal to the original payment, and if so she sends the requested preimages to Bob. At this stage, Bob can accept all the partial payments by releasing his own preimage together with Alice's preimages. As in the case of single-payment scheme, Bob's preimage can be treated as a proof of payment.

```
HTLC:{
  Vout: [{
    value: <payment>
    scriptPubKey:
      IF
        HASH160 <Hash_Bob>
        EQUALVERIFY <PK1>
      ELSE
        <delay> CSV DROP <PK2>
      ENDIF
      CHECKSIG
  }]
}
```

(a) HTLC

```
H2TLC:{
  Vout: [{
    value: <payment>
    scriptPubKey:
      IF
        HASH160 <Hash_Alice>
        EQUALVERIFY
        HASH160 <Hash_Bob>
        EQUALVERIFY <PK1>
      ELSE
        <delay> CSV DROP <PK2>
      ENDIF
      CHECKSIG
  }]
}
```

(b) H^{2}TLC

Figure 1: HTLC and H^{2}TLC in Bitcoin Script Pseudocode. PK1 and PK2 are Bob's and Alice's public keys, respectively.

4.1 Procedure

Spear makes a payment in four steps:

- **Step 1:** Alice receives an invoice from Bob through an out-of-band channel. The invoice includes the amount F that Alice has to pay and a hash digest h_b[1]. We remark that, unlike Boomerang, Alice and Bob are not required to agree on how F will be divided into partial payments; Spear allows Alice to divide F to any number of partial payments. Moreover, Spear allows Alice to set uneven partial payments, as explained in the next step.
- **Step 2:** Alice selects k payment paths to Bob. For each path, she sets an amount of partial payment, and a unique hash digest by applying a cryptographic hash function to a secret unique to the path. She then initiates the transfer of the partial payments all together. Note that, in Spear, Alice can choose any number of paths, and any amount for the partial payments. For example, suppose F = \$8. Alice can set k to 12 and send \$1 on each of these 12 paths. Or, she may select seven paths (i.e. $k = 7$), send \$4 on the first path and \$2 on each of the remaining six paths.
- **Step 3:** Using the out of band channel, Bob informs Alice of the partial payments he has received and request preimages to claim the payment. In response, Alice will reveal a subset of her preimages to Bob. To prevent Bob from overdrawing, Alice makes sure that the sum of the partial payments corresponding to the released preimages is equal to F.
- **Step 4:** Let $\mathcal{P}$ be the set of partial payments whose preimages have been released by Alice in Step 3. In Step 4, Bob claims all the partial payments in $\mathcal{P}$ if
  (1) The sum of payments in $\mathcal{P}$ is at least F, and
  (2) Bob has enough time to claim all the payments in $\mathcal{P}$.
  While claiming the partial payments, Bob cancels any received redundant payment.

Figure 2 illustrates Steps 2 and 4 of the above procedure. Note that these steps are basically equivalent to applying the conventional single-payment scheme multiple times, with the only exception that each partial payment uses H^{2}TLC instead of HTLC. In particular,

[1]Bob generates h_b by first drawing a secret (preimage), and then applying a cryptographic hash function to it.

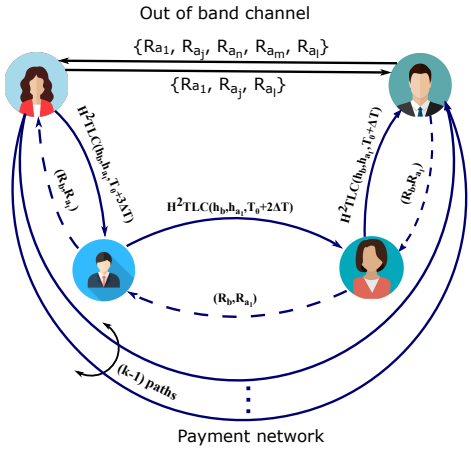

Figure 2: Making a payment in Spear.

notice that the timeouts on each path follows those of the single-payment scheme. Also, notice that Steps 1 and 3 each require only a single round of communication over an out-of-band channel.

4.2 Security Guarantees

As mentioned earlier, Spear prevents Bob from ever overdrawing. The following proposition captures this essential property of Spear.

Proposition 4.1. *Bob cannot overdraw if Alice follows the protocol.*

Proof. To claim a partial payment, Bob has to release two preimages. Since one of the two preimages is only known by Alice and is unique per partial payment, the only way Bob can claim a partial payment is to know Alice's preimage for the partial payment. Therefore, the maximum amount Bob can ever claim is limited to the sum of the partial payments whose preimages have been released by Alice in Step 3. This sum is equal to F according to Step 3 of the protocol. □

As mentioned earlier, Bob's secret preimage acts as a proof of payment, hence must remain secret until Bob receives the full payment. Spear guarantees this as stated in the next proposition.

Proposition 4.2. *If Bob follows the protocol, then Alice will know Bob's secret preimage only if Bob receives the full payment.*

Proof. According to Step 4 of the Spear procedure, Bob does not claim any partial payments if he is not guaranteed to receive at least an amount of F. In other words, Bob will release his secret preimage only if he is guaranteed a total payment of at least F. □

The following corollary is a direct consequence of Propositions 4.1 and 4.2.

Corollary 4.3. Alice can use Bob's secret preimage as a proof that she has paid Bob.

4.3 Implementation

Implementing Spear is relatively simple. In Spear, each partial payment is handled similar to the conventional single-payment scheme with only one exception: the transfer of payment on channels must be conditioned on the release of two preimages instead of one. We emphasize that, in Spear, all other parameters such as timeouts remain the same as the conventional single-payment scheme.

Consider any channel on a partial payment path. Suppose this channel is between nodes C and D. According to BOLT[2] [1], node C must send an `update-add-htlc` message to node D in order to create an HTLC. In Spear, however, node C needs to create an H^{2}TLC instead of an HTLC. For this, C must send an `update-add-H`^{2}`TLC` message to node D. As shown in Figure 3, `update-add-H`^{2}`TLC` is basically an `update-add-htlc` message with an additional field, which carries Alice's hash digest. The size of an `update-add-htlc` message is 1450 Bytes. Since `update-add-H`^{2}`TLC` carries an extra element (i.e., a hash digest of size 32 bytes) its size is 1482 bytes, which is about 2% larger than the size of an `update-add-htlc` message.

1. type: 128 (update_add_h^{2}tlc)
2. data:
   - [channel_id:channel_id]
   - [u64:id]
   - [u64:amount_msat]
   - [sha256:payment_hash]
   - **[sha256:sender_payment_hash]**
   - [u32:cltv_expiry]
   - [1366*byte:onion_routing_packet]

Figure 3: Update-add-h^{2}tlc message. Note that this is the same as the update-add-htlc message except the additional bolded line.

5 SPEAR VERSUS BOOMERANG

In [3], the authors devise Boomerang, and show that the latency of transfers reduces and the throughput increases when redundant payment paths are added. Our work is motivated by this positive result and aims to improve it through the design of Spear. To evaluate the improvement, in the following, we compare Spear and Boomerang against several factors, including payment latency, implementation complexity, computational overhead, and required liquidity and timeouts.

5.1 Latency

The main objective of both Spear and Boomerang is to reduce the payment latency, that is the time needed to complete the payment process. Therefore, it is interesting to first see how these two methods compare to each other in terms of latency.

Both Spear and Boomerang use two types of exchanges: 1) exchanges over the Lightning Network, and 2) exchanges over an

[2]BOLT (Basis of Lightning Technology) is the standardized technical specification for the implementation of the Lightning Network.

out-of-band channel. The main advantage of Spear over Boomerang with regards to latency is that it needs two exchanges of the former type while Boomerang requires three: both Spear and Boomerang use the first exchange to forward Alice's partial payments to Bob, and the second exchange for Bob to release his secret preimage. At the end of the second exchange, the transfer of payment is complete in Spear, while Boomerang requires an additional exchange so Alice can free up liquidity. As will be explained, exchanges over the Lightning Network are considerably slower than exchanges over an out-of-band channel. Therefore, one can expect Boomerang to be about 50% slower than Spear (as it requires three exchanges over the Lightning Network as opposed to two). In the following, we analyze this claim.

For a fair comparison, let us assume that both Spear and Boomerang use the same set of paths and the same amount of partial payments on these paths.

Proposition 5.1. *Let δ denote the average round-trip time between two nodes that are connected with a channel, γ denote he average round-trip time between Alice and Bob, and l denote the length of the longest path over which a partial payment is transferred from Alice to Bob.*
Then, the average payment latency of Boomerang and Spear can be estimated as $6l \cdot \delta + \gamma$ and $4l \cdot \delta + 2\gamma$, respectively.

Proof. To estimate the latency of the two payment protocols, we break their process into sequential steps, and then estimate the time they need for each step. In the first step, both processes require a single round of communication between Alice and Bob over an out-of-band channel. Spear requires this step so Alice can obtain the hash digest from Bob, while Boomerang requires this step so 1) Alice can obtain the polynomial coefficients from Bob, and 2) Alice and Bob can agree on the number of partial payments. By definition of γ, this step requires γ seconds.

In the second step, both protocols transfer partial payments from Alice to Bob. All these transfers occur in parallel, hence the time needed for this step can be estimated by the time needed for the partial payment to go through the longest path. Let l be the length of the longest path whose partial payment is accepted by Bob. The partial payment on this path goes through l channels, sequentially. At each channel, the two "channel holders" must create new commitment transactions [25]. To perform this, as illustrated in Figure 4, first the two users agree to make a new commitment. Then each user generates a new commitment transaction (CT). Since the commitment transaction is revocable, users must sign and exchange Revocable Delivery Transaction (RDTX). In addition, they must cancel the previous commitment transaction by signing a Breach Remedy Transactions (BRTX), and exchanging these transactions. This process, as shown in Figure 5, requires two rounds of communications between the two channel holders, hence requires about 2δ seconds. Since this process is repeated sequentially over l channels, the second step in both Spear and Boomerang require about $2\delta \cdot l$ seconds[3].

In the next step, Spear requires a single round of communication between Alice and Bob over the out-of-band channel. In this step, Bob informs Alice about the set of receive partial payment. In response, Alice sends Bob the corresponding preimages. This step therefore needs γ seconds. Note that Boomerang does not require this step. In the next step, both Spear and Boomerang require Bob to release his secret and claim the partial payments. Similar to the second step, this step requires about $2\delta \cdot l$ seconds in both protocols. In the final step, which is only required by Boomerang, Alice renounces the option to react and frees up the liquidity. Similar to the previous step, this step requires $2\delta \cdot l$ seconds.

Adding delays of all steps for the two protocols, we get that Spear requires about $4l \cdot \delta + 2\gamma$ seconds, and Boomerang needs about $6l \cdot \delta + \gamma$ seconds.

□

Corollary 5.2. The payment latency of Boomerang is up to 50% higher than that of Spear.

Proof. By Proposition 5.1, the ratio of the payment latency of Boomerang over the payment latency of Spear can be approximated as

$$\frac{6l \cdot \delta + \gamma}{4l \cdot \delta + 2\gamma},$$

which is at most equal to 1.5. □

Example 3. Suppose Alice is paying Bob for a cup of coffee she is purchasing at Bob's coffee shop. Assuming that γ is much smaller than δ (e.g., Alice and Bob are connected to the same network), we get

$$\frac{6l \cdot \delta + \gamma}{4l \cdot \delta + 2\gamma} \approx \frac{6l \cdot \delta}{4l \cdot \delta} = 1.5.$$

On the other hand, if the connection between Alice and Bob has a round-trip time of δ, then, assuming $l = 4$, we get

$$\frac{6l \cdot \delta + \gamma}{4l \cdot \delta + 2\gamma} = \frac{6l \cdot \delta + \delta}{4l \cdot \delta + 2\delta} = \frac{25\delta}{18\delta} \approx 1.39.$$

5.2 Implementation

Boomerang's implementation requires a new opcode or use of adaptor signatures [3]. Implementing Spear, on the other hand, is straightforward; we mainly need to replace HTLC with H^{2}TLC, which is identical to HTLC, except it conditions the payment on release of two preimages instead of one.

An advantage of Boomerang over Spear is that it does not need a duplex channel. This is because Bob can set the number of partial payments on his own, and include this information in the invoice. If Alice and Bob need to agree on the number of partial payments, however, a duplex channel is needed.

5.3 Locktime

As shown in Figure 6, for a payment path of length l (i.e., a path with l channels), Spear requires a maximum locktime of $l \cdot \Delta$, where Δ is the time set so that a node can forcefully redeem a fund on chain[4]. This is identical to the maximum locktime set in the conventional single-path payment scheme. Boomerang, on the other hands, requires the maximum locktime of $2 \cdot l \cdot \Delta$ [3], which is

[3] In this step, Bob cancels the redundant partial payments at the same time that he accepts the other partial payments. The time needed to finish this step is, therefore, more accurately captured as $2\delta \cdot \max(l, l')$, where l' is the length of the longest redundant path.

[4] Values of Δ are typically either 40 blocks or 144 blocks [17].

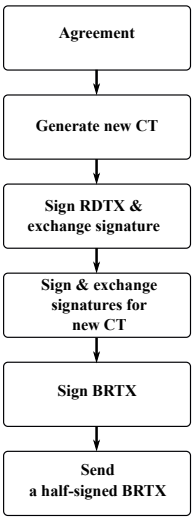

Figure 4: The process for creating new commitment transactions

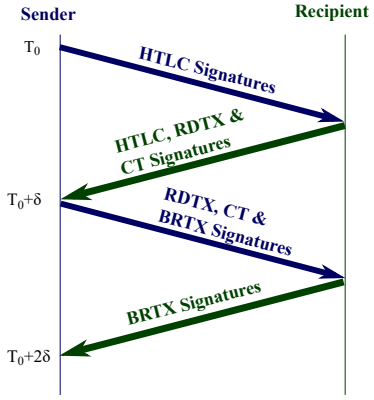

Figure 5: The process of creating an HTLC.

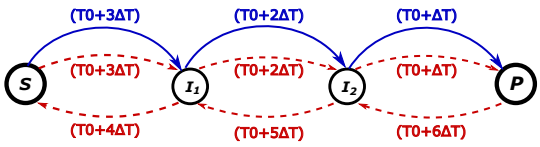

Figure 6: Transfer of a partial payment from S to P. The red dashed lines, and solid blue lines show locktime in Boomerang and Spear, respectively.

twice what is needed in Spear. Therefore, in the worst case (e.g., as a result of a griefing attack) funds can be locked for a longer period of time in Boomerang than in Spear.

5.4 Computational overhead

The computation that Spear imposes on Bob is limited to computing a hash function per partial payment. In Boomerang, on the other hand, Bob has to compute a finite field exponentiation per partial payment, with each exponential requiring at least q field multiplications, where q is the order of the finite field in bits. Assuming that a field multiplication is of the same order of computation as a single hash computation, the computational overhead imposed by Boomerang is at least a factor of f of that of Spear[5]. For example, assuming that Boomerang uses a finite field of order 256 bits, we get that the computational overhead that Boomerang imposes on Bob is at least two orders of magnitude higher than what Spear imposes.

As for Alice, Boomerang requires at least v^2 field multiplications per partial payment, where v is the maximum number of partial payments that Bob can accept (Proposition 5.3). Spear, however, requires a single hash computation per partial payment. Therefore, Boomerang imposes more computational overhead on Alice than Spear does. In particular, note that Boomerang's required computation per partial payment grows quadratically with the number of partial payments, while in Spear, the amount of computation per partial payment is constant.

Proposition 5.3. *In Boomerang, Alice requires to compute at least v^2 field multiplications per partial payment.*

Proof. In Boomerang, Alice has to compute

$$\forall i \in \{1, 2, \ldots, v, \ldots, w\} \quad H(P_i) = \Pi_{j=0}^{v} H(\alpha_j)^{(i^j)}, \tag{3}$$

where w is the total number of partial payments, v is the maximum number of partial payments that Bob can accept, and $H(\alpha_j)$ are digests that Bob provides Alice. By (3), Alice needs to compute $w \cdot v$ exponentiations. Note that the largest exponent in (3) is w^v. Therefore, by Yao's result [31], Alice needs to perform at least

$$(w \cdot v + o(1)) \log(w^v) / \log\log(w^v) \sim w \cdot v^2$$

field multiplications. Therefore, the minimum number of field operations needed by Alice is $(w \cdot v^2)/w = v^2$. □

5.5 Flexibility

In Boomerang, Alice and Bob must agree (out-of-band) to partition the payment into v partial payments. In addition, Boomerang inherently requires all the partial payments to be of the same value. In Spear, on the other hand, Alice can decide on her own how to divide the payment into partial payments. Moreover, Spear allows Alice to divide the payment into unequal partial payments. This provides high flexibility to Alice to handle the payment. For instance, if Alice has a prior knowledge that a certain path (e.g. a direct channel to Bob) can carry a large portion of the total payment, then she can use this path to transfer a large partial payment to Bob in a single partial payment. Or, Alice may try to transfer larger partial payments on shorter paths than on larger paths, since a partial payment is more likely to go through a shorter path than a larger one (e.g., see Corollary 3.1.1).

[5]This factor can be made smaller (in the best case to $\frac{f}{\log f}$) at the expense of more storage.

Table 1: Comparison of AMP, Boomerang, and Spear.

Multi-path payment model	AMP	Boomerang	Spear
Supports redundant payments?	No	Yes	Yes
proof of payment in invoice?	No	Yes	Yes
Requires a duplex channel?	No	No [6]	Yes
Allows heterogeneous partial payments?	Yes	No	Yes
Latency	$4l \cdot \delta + \gamma$	$6l \cdot \delta + \gamma$	$4l \cdot \delta + 2\gamma$
Implementation	Available in LND [12]	Requires a new opcode or use of adaptor signatures	Replace HTLC with H^2TLC
Maximum Locktime	$l \cdot \Delta$	$2l \cdot \Delta$	$l \cdot \Delta$
Computation	hash function	field exponentiation	hash function

5.6 Intermediate node's misbehaviour

In the Lightning Network, an intermediate node can stall a payment for hours by accepting an incoming payment and not forwarding the payment to the next node. In the single-path payment scheme, Alice has to wait for a long period of time (until this payment is canceled) before she can make a second attempt to pay Bob. HTLC-based multi-path payment methods are also vulnerable; if a single partial payment is stalled by a misbehaving node, Alice has to wait until this partial payment is canceled before she can make a second attempt. This may not be desired, as Alice may prefer to pay Bob as soon as possible, even if one or more partial payments are stalled/locked for a period of time.

Multi-path payment methods with redundant paths such as Spear and Boomerang naturally mitigate this issue. It is because these methods, by definition, support redundant paths, and as long as Bob receives enough number of partial payments, the payment can go through. However, there is a difference between Spear and Boomerang when it comes to freeing up liquidity.

In Boomerang, if a misbehaving node stalls a single partial payment on path P then Alice may not free up the liquidity on other paths until the partial payment on P is canceled. It is because, Alice cannot distinguish between the following two scenarios: 1) an intermediate node on path P is stalling the partial payment; 2) Bob is holding on to the partial payment on P so he can claim it later. Therefore, Alice may not renounce the reverse components of Boomerang contracts to free up their liquidity, as there is a chance that Bob overdraws later, at which point Alice can revert the partial payments. In Spear, on the other hand, Bob can immediately free up the liquidity on all the paths except P by accepting the partial payments, and cancelling the redundant ones. Therefore, in Spear, the misbehaving node on path P can only delay the process of freeing up the liquidity of path P but not other paths.

5.7 Fees

In both Boomerang and Spear, the sum of partial payments Bob receives can be higher than the full payment. In this case, assuming Bob is honest, he will only claim a subset of the received partial payments whose sum is equal to the full payment. In Spear, Bob can go a bit further and help Alice to reduce her fees by reporting all the partial payments he receives. This allows Alice to choose the subset of partial payments whose sum of fees is minimum. In Boomerang, on the other hand, Bob does not know which subset of partial payments to accept in order to reduce Alice's fees. It is because, Bob is unaware of the fees Alice is paying on each partial payment path (unless Alice communicates this information to Bob).

6 RELATED WORK

There are three main classes of payment routing methods [7]: single-path payment [25], multi-path payment [14, 19, 27], and packet-switch routing [23, 28, 29]. In the single-path routing, Alice sends the whole payment through a single path. In multi-path payment, Alice can split her payment into partial payments and transfer them through different paths. Lastly, in the packet-switch routing, Alice splits the payment into unit of payments and sends them individually.

The multi-Path Payment (MPP) method [19] is the simplest type of multi-path payment proposed for the Lightning Network. In MPP, Alice splits the payment into partial payments and sends them through different paths. All these partial payments are conditioned on the disclosure of a single preimage known by Bob. When Bob receives all the partial payments, he can accept/settle all these payments by disclosing his preimage.

Another existing multi-path payment method for the Lightning Network is Atomic Multi-Path Payment (AMP) proposed in [21]. In AMP, Alice uses a "base preimage" for the payment, from which she derives payment preimages of partial payments. As in MPP, Alice splits the payment into partial payments, and transfers these payments through different paths. However, in AMP, Bob needs to receive all the partial payments in order to construct the preimages and claim the partial payments. Therefore, unlike MPP, in AMP Bob cannot claim any partial payment before he receives all the partial payments.

[6] Boomerang needs a duplex channel if Alice and Bob should agree on the number of partial payments.

In both MPP and AMP, the whole payment fails/delays if any of the partial payments fails/delays. Packet-switch routing methods [6, 9, 18, 26, 27, 30] can alleviate this to some extent. For example, in Ethna [5], and Interdimensional Speedy Murmurs [6] protocol and its extension [27], in addition to Alice, intermediate nodes are able to split a payment. Furthermore, these protocols allow intermediate users to select the next user on the path. However, as in MPP and AMP, the whole payment fails/delays if any single partial payment fails/delays. In addition, packet-switch routing methods may require a high number HTLCs. This can put pressure on the network as each channel can support only a limited number of (unsettled) HTLCs.

One can consider a fourth class of payment methods: multi-path payments with redundancy. As far as the authors know, Boomerang [3] is the only existing method that falls within this class. In this work, we propose Spear, the second method within this class. As shown in [3] and this work, these methods have a great potential in improving the performance of multi-path payment.

7 CONCLUSION

In this work, we showed that a payee can significantly improve the success probability of payments by sending redundant partial payments. We proposed Spear, a simple multi-path payment method with redundancy, and showed how it can be implemented in the Lightning Network. Moreover, we compared Spear with Boomerang, the only existing multi-path payment with redundancy in the literature. Our comparison results show that Spear has several advantages over Boomerang, including lower delay, ease of implementation, and lower computational requirement. The approach used in Spear can be used in other payment methods, such as packet-switch routing methods, for further performance improvement.

REFERENCES

[1] BOLT 2. 2019. Peer Protocol for Channel Management. https://github.com/lightningnetwork/lightning-rfc/blob/master/02-peer-protocol.md. Last Accessed: 2020-10-26.

[2] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timon, and Pieter Wuille. 2014. Enabling blockchain innovations with pegged sidechains. http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains.

[3] Vivek Kumar Bagaria, Joachim Neu, and David Tse. 2020. Boomerang: Redundancy Improves Latency and Throughput in Payment-Channel Networks. In *Proceedings of the 24th International Conference on Financial Cryptography and Data Security FC* (Kota Kinabalu, Malaysia) *(Lecture Notes in Computer Science, Vol. 12059)*. Springer, 304–324. https://doi.org/10.1007/978-3-030-51280-4_17

[4] Christian Decker and Roger Wattenhofer. 2015. A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. In *Proceedings of the 17th International Symposium on Stabilization, Safety, and Security of Distributed Systems SSS* (Edmonton, AB, Canada) *(Lecture Notes in Computer Science, Vol. 9212)*. Springer, 3–18. https://doi.org/10.1007/978-3-319-21741-3_1

[5] Stefan Dziembowski and Pawel Kedzior. 2020. Ethna: Channel Network with Dynamic Internal Payment Splitting. *IACR Cryptol. ePrint Arch.* 2020 (2020), 166.

[6] Lisa Eckey, Sebastian Faust, Kristina Hostáková, and Stefanie Roos. 2020. Splitting Payments Locally While Routing Interdimensionally. *IACR Cryptol. ePrint Arch.* (2020), 555. https://eprint.iacr.org/2020/555

[7] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. 2019. SoK: Off The Chain Transactions. *IACR Cryptol. ePrint Arch.* (2019), 360. https://eprint.iacr.org/2019/360

[8] Mike Hearn. 2013. Micro-payment channels implementation now in bitcoinj. https://bitcointalk.org/.

[9] Philipp Hoenisch and Ingo Weber. 2018. AODV-Based Routing for Payment Channel Networks. In *Proceedings of the First International Conference on BlockchainICBC, Held as Part of the Services Conference Federation, SCF* (Seattle, WA, USA) *(Lecture Notes in Computer Science, Vol. 10974)*. Springer, 107–124. https://doi.org/10.1007/978-3-319-94478-4_8

[10] Rami Khalil, Arthur Gervais, and Guillaume Felley. 2018. Nocust-a securely scalable commit-chain. *IACR Cryptol. ePrint Arch.* (2018), 642.

[11] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *Proceedings of the IEEE Symposium on Security and Privacy, SP* (San Francisco, California, USA). IEEE Computer Society, 583–598. https://doi.org/10.1109/SP.2018.000-5

[12] LND. 2020. Lightning Network Daemon. https://github.com/lightningnetwork/lnd. Last Accessed: 2020-08-19.

[13] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A Secure Sharding Protocol For Open Blockchains. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 17–30. https://doi.org/10.1145/2976749.2978389

[14] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. 2017. SilentWhispers: Enforcing Security and Privacy in Decentralized Credit Networks. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society. https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/silentwhispers-enforcing-security-and-privacy-decentralized-credit-networks/

[15] Robert J. McEliece and Dilip V. Sarwate. 1981. On sharing secrets and Reed-Solomon codes. *Commun. ACM* 24, 9 (1981), 583–584. https://doi.org/10.1145/358746.358762

[16] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. 2019. Sprites and State Channels: Payment Networks that Go Faster Than Lightning. In *Proceedings of the 23rd International Conference on Financial Cryptography and Data Security FC (Lecture Notes in Computer Science, Vol. 11598)*. Springer, 508–526. https://doi.org/10.1007/978-3-030-32101-7_30

[17] Ayelet Mizrahi and Aviv Zohar. 2020. Congestion attacks in payment channel networks. *arXiv preprint arXiv:2002.06564* (2020).

[18] Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Kim Pecina. 2015. Privacy Preserving Payments in Credit Networks: Enabling trust with privacy in online marketplaces. In *Proceedings of the 22nd Annual Network and Distributed System Security Symposium, NDSS* (San Diego, California, USA). The Internet Society.

[19] MPP. 2020. routing: multi-part mpp send. https://github.com/lightningnetwork/lnd/pull/3967. Last Accessed: 2020-11-12.

[20] Raiden Network. 2018. Fast, cheap, scalable token transfers for Ethereum. https://raiden.network/. Last Accessed: 2020-08-19.

[21] Olaoluwa Osuntokun. 2018. AMP: Atomic Multi-Path Payments over Lightning. https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html.

[22] David A. Patterson and John L. Hennessy. 2017. *Computer Organization and Design - The Hardware/Software Interface (RISC-V Edition)*. The Morgan Kaufmann Series in Computer Architecture and Design.

[23] Dmytro Piatkivskyi and Mariusz Nowostawski. 2018. Split Payments in Payment Networks. In *Proceedings of the International Workshops on Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS* (Barcelona, Spain) *(Lecture Notes in Computer Science, Vol. 11025)*. Springer, 67–75. https://doi.org/10.1007/978-3-030-00305-0_5

[24] Joseph Poon and Vitalik Buterin. 2017. Plasma: Scalable autonomous smart contracts. *White paper* (2017), 1–47.

[25] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments.

[26] Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. 2016. Flare: An approach to routing in lightning network. *White Paper* (2016).

[27] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. 2018. Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium, NDSS* (San Diego, California, USA). The Internet Society.

[28] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia C. Fanti, and Pramod Viswanath. 2018. Routing Cryptocurrency with the Spider Network. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks,HotNets* (Redmond, WA, USA). ACM, 29–35. https://doi.org/10.1145/3286062.3286067

[29] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathleen Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Giulia C. Fanti, and Mohammad Alizadeh. 2020. High Throughput Cryptocurrency Routing in Payment Channel Networks. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI* (Santa Clara, CA, USA). USENIX Association, 777–796.

[30] Bimal Viswanath, Mainack Mondal, P. Krishna Gummadi, Alan Mislove, and Ansley Post. 2012. Canal: scaling social network-based Sybil tolerance schemes. In *Proceedings of the 7th European Conference on Computer Systems* (Bern, Switzerland). ACM, 309–322. https://doi.org/10.1145/2168836.2168867

[31] Andrew Chi-Chih Yao. 1976. On the Evaluation of Powers. *SIAM J. Comput.* 5, 1 (1976), 100–103. https://doi.org/10.1137/0205008