Grief-free Atomic Swaps

Tejaswi Nadahalli
ETH Zürich

Majid Khabbazian
University of Alberta

Roger Wattenhofer
ETH Zürich

***Abstract*—Atomic Swaps enable exchanging crypto-assets without trusting a third party. To enable these swaps, both parties lock funds and let their counterparty withdraw them in exchange for a secret. This leads to the so-called *griefing attack*, or the emergence of an American Call option, where one party stops participating in the swap, thereby making their counterparty wait for a timelock to expire before they can withdraw their funds. The standard way to mitigate this attack is to make the attacker pay a premium for the emerging American Call option. In these premium-paying approaches, the premium itself ends up being locked for possibly an even longer duration than the swap amount itself. We propose a new Atomic Swap construction, where neither party exposes itself to a griefing attack by their counterparty. Notably, unlike previous constructions, ours can be implemented in Bitcoin as is. Our construction also takes fewer on-chain transactions and has a lower worst-case timelock.**

***Index Terms*—Bitcoin, Atomic Swaps, Griefing**

I. Introduction

Before Bitcoin, there was considerable research on the *Fair Exchange Problem*, with its associated impossibility results [1], [2], [3], [4]. Many of these results came about in the quest to remove the *trusted third party* when two parties want to exchange different assets of equal value. Bitcoin created a different kind of trusted third party. Here, both parties trust the Bitcoin blockchain as an arbiter for dispute resolution and as a tamperproof public bulletin board. Fair Exchange, in the Bitcoin setting, is known as the Atomic Swap. It is atomic in the sense that a swap of assets between two parties either goes through fully or not at all. Atomic Swaps were perhaps the first non-trivial smart contracts designed to work on blockchains. Tier Nolan's classic swap (TN-swap, from here on) was discussed on the BitoinTalk forum in 2013 [5]. The TN-swap is not atomic from a transaction perspective. The swap requires 4 transactions: 2 HTLCs + (2 redeems or 2 refunds).[1] The atomicity is from a higher abstraction of the swap of assets – either the swap goes through, and both parties end up with the assets they desire, or it does not, and both parties retain their original assets. These assets could even be on different blockchains, e.g., Bitcoin and Litecoin. If it were a cross-chain atomic swap, both blockchains must be compatible with the primitives used in the swap protocol.

Atomic swaps could involve various assets: say, someone wants to buy a Sudoku solution for some price. However, these generally involve more complex protocols that convert assets into information that can be transferred on a public blockchain. In the case of a Sudoku, a symmetric key is used to encrypt the solution, and this key is atomically swapped for monetary value, while the encrypted blob is sent off-chain, as seen in ZKCP [7]. This kind of swap has one of the assets not being scrutinizable on the blockchain and hence has to rely on more complex Zero-Knowledge proofs to convince the buyer that the key encrypts a correct solution. We want to look at a more straightforward class of atomic swaps where one can scrutinize the actual asset being swapped on the blockchain, and the buyer or the swap initiator doesn't need any other data beyond the blockchain. In the base case, these swaps involve the native cryptocurrency of the blockchain(s). In the single blockchain setting, swapping coins of equal value between Alice and Bob can improve both their privacy, as proposed in Coinswap [8].

TN-swap, which is formally defined in Section III-A, is atomic but not fair. There are steps in the swap where either Alice or Bob can abort the protocol and put their counterparty at a disadvantage. The counterparty does not lose monetary value (it is an atomic swap, after all) but is either made to wait before they get their asset back or might lose blockchain fees by making extra transactions and such. We refer to the waiting part of this problem as *griefing*. Protocols like Arwen [9] rely on one of the counterparties of the atomic swap caring about protecting their reputation. A few proposals have been made to reduce griefing, but they all involve smart contracts that have access to global state storage. These smart contracts look up the swap state and proceed according to how the swap has gone so far. Some of these proposals are: Fairswap [10], Optiswap [11], Han-Lin-Yu swap (HLY-swap) [12], and Xue-Herlihy swap (XH-swap) [13]. All these rely on smart contracts and are not compatible with Bitcoin natively. The former two optimize for the *optimstic case*, where the swap is efficient to execute if it goes as expected. In the *pessimistic case*, when the swap does not go through, a more complex dispute resolution protocol is invoked. The latter two are more focused on avoiding our griefing problem and solve it by getting the advantaged party paying a premium[2] to the disadvantaged party. The HLY-swap paper draws parallels between the atomic swap and an American Call Option from traditional finance. In traditional finance, these options are made fair by getting the disadvantaged party to sell the option itself to the advantaged party. The price at which this option is sold is called the option premium. As the advantaged party pays this premium upfront to the disadvantaged party, their privilege to abort the swap is compensated for. Unfortunately,

[1] HTLC stands for Hash Timelock Contracts [6], and the redeem and refund transactions are part of the HTLC specification.

978-1-6654-9538-7/22/$31.00 ©2022 IEEE

[2]

Bitcoin's stack-based execution environment does not allow access to external state storage, and these swaps cannot be directly implemented in Bitcoin natively. Both these swaps can be implemented in Bitcoin if a new opcode is added to it. Done that way, HLY-Swap uses the premium on one side of the swap but allows griefing on the other side. XH-swap avoids griefing on both sides of the swap but allows the premiums themselves to be griefed. Contrary to the claim in the XH-swap paper that griefing cannot be avoided, we present an atomic swap construction without griefing, which can also be implemented in Bitcoin with no changes to Bitcoin itself. Our protocol is also more efficient regarding the number of transactions and the worst-case timelock for which funds are locked.

II. System Model

Our system model is based on Bitcoin, with its UTXO (Unspent Transaction Outputs) model. We require primitives like hashes, timelocks, and signatures. Furthermore, we make the following assumptions about the system.

- Time proceeds as blocks, and each block is separated by a constant and known unit of real-world time.
- All users are online and through the public blockchain, know if specific transactions are confirmed.
- Transactions have constant fees, which are independent of the amounts involved in the transactions.

A. Atomic Swap Specification

The Atomic Swap specification consists of the following:

- Two users: Alice and Bob, who want to swap their coins P_a and P_b with each other. These could be on different blockchains or the same blockchain. We call this the principal amount.
- A sequence of n transactions S_{all}, made up of individual transactions T_0, T_1, T_2, …, T_{n-1}, out of which a subset S_{conf} get confirmed on the blockchain.
- At the end of S_{conf}, **only one** of the following is true:
  - Successful swap: Alice has value equivalent to P_b and Bob has value equivalent to P_a. We call this set $S \subset S_{all}$. Note that there is typically a single subset of S_{all} that makes a successful swap.
  - Unsuccessful swap: Alice has value equivalent to P_a and Bob has value equivalent to P_b. We call this set $F \subset S_{all}$. Note that there could be many subsets $F_1, F_2, \ldots$ that make up different failure scenarios of the swap.

B. Notation

We recall that in the UTXO model, every transaction consumes unspent outputs of previous transactions and creates the next set of unspent outputs for other transactions to spend.[3] An unspent transaction output (UTXO) contains the coin value in question and a set of locking conditions. Unlocking conditions will come from the transaction that spends this UTXO. A UTXO can be locked with various primitives like digital signatures, timelocks, knowledge of preimages of hashes, basic arithmetic, and such. Note that the locking conditions and their corresponding unlocking conditions are evaluated together on the stack, and there is no other external input available during this evaluation. Not having access to external state data makes the Bitcoin model *stateless*. Being stateless in this specific way makes designing smart contracts harder.

In this paper, we use the transaction/predicate notation for UTXO based transactions from the Cerberus Channels paper [14]. We let $o = (x \mid P)$ to represent a UTXO that holds value x and lists a predicate P that locks or unlocks this UTXO. Predicate P can be a base predicate (see list below) or a combination of base predicates with $\vee$ (*OR*) or $\wedge$ (*AND*) operators.

- σ_a: Signature that matches[4] the public key A.
- $s \ni h(s) = H_s$: The spending transaction needs to provide a preimage s whose hashed value is H_s.
- Δ_k: A timelock of k blocks needs to elapse to unlock the spending transaction.

A transaction is a mapping from a set of past UTXO's to a set future UTXO's, and can be represented as:

$$T_i = [o_j, o_k, \ldots] \mapsto [o_i^1, o_i^2, \ldots]$$

where T_i consumes past UTXO's o_j, o_k, …. to produce future UTXO's $\{o_i^1, o_i^2, \ldots\}$. Predicates that apprear on the left side of a transaction unlock the UTXOs in question, and those that appear on the right side lock the newly created UTXO's. An example transaction would look like:

$$T_i = [(2 \mid \sigma_a), \underbrace{(1 \mid \Delta_{10}), (3 \mid s_x)}_{T_j}] \mapsto [(6 \mid \sigma_b \wedge H_{s_y})]$$

T_i is spending 3 UTXO's by providing a signature σ_a for A (Alice), waiting for time Δ_{10} (10 blocks), and a preimage s_x such that the hash $h(s_x)$ was used to lock the 3rd UTXO that T_i is spending. T_i itself creates a new UTXO that has the coin value of 6, and can be spent by providing a signature for B (Bob) and a preimage s_y such that the hash $h(s_y) = H_{s_y}$. Additionally, the two spent UTXO's $(1 \mid \Delta_{10}), (3 \mid s_x)$ were created in a previous transaction T_j. The UTXO creating transaction (in the underbrace) is shown only if it's relevant to the protocol. In the above case, the UTXO $(2|\sigma_a)$ doesn't show a source UTXO under it, and can be assumed that the transacton from where Alice got her 2 bitcoins doesn't matter in this setting.

III. Atomic Swaps: Prior Work

In this section, we formally define Tier Nolan's classic atomic swap [5] and two other sophisticated swap designs that avoid griefing to some extent but do not eliminate it. This formal treatment of these swaps reveals the scenarios where griefing happens, how premiums prevent griefing, and how premiums themselves can be griefed.

[3] Coinbase transactions are unique transactions that do not have inputs and create new coins.

[4] Bitcoin uses SIGHASH flags to control which part of a transaction is signed by whom. For simplicity, we assume what is being signed is clear from the context.

A. Tier Nolan Atomic Swap

In Tier Nolan's classic swap (TN-swap), Alice locks P_a with a timelock (refunding P_a back to her) and a hashlock (so Bob can redeem P_a). Bob locks P_b symmetrically but with a lower value for the timelock. If one of the parties aborts, the other party can wait for their timelock to expire and refund their principals back to themselves. If neither party aborts, the swap completes with both parties redeeming the principals due to them. The blockchain acts as a public bulletin board that communicates the secret preimage of the hashlock from Alice to Bob. Alice's timelock is always longer than Bob's to account for Alice's head start in the protocol and knowing the preimage of the hashlock, which enables her to finish her side of the swap first. We use the word *refund* when a party's principal comes back to them after a swap is abandoned, and the word *redeem* when a party can complete a swap and get the counterparty's principal. The entire set of transactions that make up the TN-swap can be defined succinctly in our notation as shown in Figure 1.

$S_{all} = \{T_0, T_1, T_2, T_3, T_4, T_5\}$
$T_0 = [(P_a|\sigma_a)] \mapsto [(P_a|(\sigma_a \wedge \Delta_2) \vee (\sigma_b \wedge H_s))]$
$T_1 = [(P_b|\sigma_b)] \mapsto [(P_b|(\sigma_a \wedge H_s) \vee (\sigma_b \wedge \Delta_1))]$
$T_2 = \underbrace{[(P_b|(\sigma_a \wedge s))]}_{T_1} \mapsto [(P_b|\sigma_a)]$
$T_3 = \underbrace{[(P_a|(\sigma_b \wedge s))]}_{T_0} \mapsto [(P_a|\sigma_b)]$
$T_4 = \underbrace{[(P_b|(\sigma_b \wedge \Delta_1))]}_{T_1} \mapsto [(P_b|(\sigma_b)]$
$T_5 = \underbrace{[(P_a|(\sigma_a \wedge \Delta_2))]}_{T_1} \mapsto [(P_a|(\sigma_a)]$
$S = \{T_0, T_1, T_2, T_3\}$
$F_1 = \{T_0, T_5\}$
[Bob aborts before committing P_b. Alice has to wait for Δ_2, and gets no compensation]
$F_2 = \{T_0, T_1, T_4\}$
[Alice aborts before redeeming P_b. Bob has to wait for Δ_1, and gets no compensation]

Figure 1: Tier Nolan Atomic Swap

Griefing: Note that Alice and Bob both have the right to abort out of the swap before it happens. If either party aborts, their counterparty is left waiting for their timelock to expire before getting their refund. If Bob aborts, Alice has to wait for Δ_2 to expire before refunding her principal P_a back to her. If Alice aborts, Bob has to wait for Δ_1 to refund his principal P_b back to him. This leads to the notion of the locked value of funds or griefing. To account for this, we add a griefing cost to each subset F_i in the atomic swap specification from Section II-A. If the griefing cost is zero for all failure subsets in the specification, we consider a protocol to be grief-free. Formally, let

$$cost = \sum f(P_i \cdot \Delta_j) \quad \forall i \in \{a, b\}, j \in \{1, 2, 3, \ldots\} \quad (1)$$

where $f(P_i \cdot \Delta_j)$ is the value of locking P_i for duration Δ_j. In the summation, a party's term $f(P_i \cdot \Delta_j)$ is introduced only if a counterparty has aborted. The timelocked value of the aborting party's principal or premium is not included in the griefing cost. TN-swap's costs for its two failure scenarios can be quantified as:

$$\begin{aligned} cost_{F_1} &= f(P_a \cdot \Delta_2) > 0 \\ cost_{F_2} &= f(P_b \cdot \Delta_1) > 0 \end{aligned} \quad (2)$$

The idea of locking up the principal amount to enable swaps seems inherent to atomic swap protocols that use sequential transactions. To lower the griefing cost of locking up the principal, atomic swaps have been proposed that offer a premium as compensation to the locking party if their counterparty aborts from the swap. This premium's value is estimated using the Cox-Ross-Rubinstein model in [12] using options pricing theory and the price volatility of the crypto-assets in question. We ignore the price volatility of the crypto assets and use a simple interest rate model to price the time value of the locked-up principal. This could be as simple as a simple interest rate, calculated by taking the product of the principal, length of the timelock, and an arbitrary interest rate r that the parties agree upon.

$$f(P_i \cdot \Delta_j) = \rho_i = P_i \cdot \Delta_j \cdot r \quad \forall i \in \{a, b\}, j \in \{1, 2, 3, \ldots\} \quad (3)$$

The total griefing cost of the protocol has to account for both the locked value of the principals and the equivalent premiums that are returned to the parties based on how the parties act during the protocol execution. Equation 1 for cost can be modified as:

$$cost = \sum f(P_i \cdot \Delta_j) - \sum \rho_i \quad \forall i \in \{a, b\}, j \in \{1, 2, 3, \ldots\} \quad (4)$$

If all the locked-up principals are compensated by corresponding premiums, $cost$ goes to zero. TN-swap's cost is strictly positive as it does not have compensatory premiums. Interestingly, in subsequent protocols we discuss, the premium ρ_i could also be locked up for some timelock Δ_j. In this case, this timelocked value of the premiums is recognized in the first term of the right-hand side of Equation 4.

B. Atomic Swaps with Premiums

We now consider two constructions that offer a premium as compensation to the party that locks up capital during the swap. The Han-Lin-Yu atomic swap (HLY-swap), introduced in [12] and the Xue-Herlihy Atomic Swap (XH-swap), introduced in [13]. Regarding the premium value itself, there are many approaches from the world of traditional finance to calculate the premium [12]. This premium has to be baked into the swap protocol - so that it can be transferred from

one party to another based on how the swap proceeds. If the blockchain in question supports stateful smart contracts, like Ethereum, this coupling between the premium and the swap can be implemented as shown in [12]. If the blockchain does not support stateful smart contracts (Bitcoin does not), both [12] and [13] suggest upgrading the scripting language of the blockchain to support it. This upgrade comes in the form of a new opcode that can scan the blockchain to see where the swapped assets ended up and then redirect the premium to that address. In other words, an opcode that can use information not available at the time of writing the contract. In Bitcoin, this opcode (called `OP_LOOKUP_OUTPUT` in [12]) requires a separate index to be maintained by the nodes. Given how Bitcoin optimizes for a smaller footprint, such a new index is unlikely to be added in the future. To analyze these swaps that need `OP_LOOKUP_OUTPUT`, we introduce two new predicate types in our notation to represent what `OP_LOOKUP_OUTPUT` does.

- $\overline{T_i.\Delta_j}$: A future transaction T_i happens before Δ_j.
- $\neg\overline{T_i.\Delta_j}$: A future transaction T_i does not happen before Δ_j.

HLY-swap: If we assume the `OP_LOOKUP_OUTPUT` opcode implemented by a combination of the above two predicates, HLY-swap can be implemented as shown in Figure 2. This swap handles the second half of the griefing problem from TN-swap, but not the first. In TN-swap, Alice can grief Bob by not redeeming his principal by broadcasting $\texttt{TN-T}_3$ (transaction T_3 from the Tier Nolan swap in Figure 1). In HLY Swap, Alice puts up a premium ρ_a in $\texttt{HLY-T}_0$, which will go back to Alice only if Alice goes along with her side of the swap in $\texttt{HLY-T}_3$. If she aborts here, Bob doesn't have the secret preimage to redeem his side of the swap and has to wait for his timelock Δ_2 to expire to get his principal back. To compensate for this grief, Bob gets to keep Alice's premium ρ_a by broadcasting $\texttt{HLY-T}_7$. This is covered in the scenario $\texttt{HLY-F}_2$.

Note that HLY-swap does not compensate Alice in case Bob aborts before committing his principal. Alice has to wait for Δ_3 before getting P_a back in $\texttt{HLY-T}_6$ and Δ_4 before getting back ρ_a in $\texttt{HLY-T}_8$ (scenario $\texttt{HLY-F}_1$). The griefing costs of HLY-swap are:

$$cost_{F_1} = f(\rho_a \cdot \Delta_4) + f(P_a \cdot \Delta_3) - \rho_a > 0 \quad (5)$$

$$cost_{F_2} = f(P_b \cdot \Delta_2) - \rho_a = 0 \quad (6)$$

As Alice aborts in failure scenario $\texttt{HLY-F}_2$, only Bob's principal P_b is included in Equation 6. Bob's timelocked value $f(P_b.\Delta_2)$ is compensated by Alice's premium ρ_a, and hence cost of failure scenario $\texttt{HLY-F}_2$ $cost_{F_2} = 0$.

As Bob aborts in failure scenario $\texttt{HLY-F}_1$, only Alice's principal P_a is included in Equation 5, and $cost_{F_1} > 0$. In fact, Alice gets extra grief here because her premium ρ_a is also locked up for Δ_4. The problem of Alice not being compensated for locking up her principal is solved in XH-swap.

$$S_{all} = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8\}$$

$$T_0 = [(\rho_a|\sigma_a)] \mapsto [(\rho_a|((\sigma_a \wedge \sigma_b \wedge \Delta_4) \wedge (\overline{T_3.\Delta_2} \vee \overline{T_5.\Delta_4} \vee \overline{T_6.\Delta_4} \vee \neg\overline{T_2.\Delta_1}))]$$

$$T_1 = [(P_a|\sigma_a)] \mapsto [(P_a|(\sigma_a \wedge \Delta_3) \vee (\sigma_b \wedge H_s))]$$

$$T_2 = [(P_b|\sigma_b)] \mapsto [(P_b|(\sigma_a \wedge H_s) \vee (\sigma_b \wedge \Delta_2))]$$

$$T_3 = [\underbrace{(P_b|(\sigma_a \wedge s))}_{T_1}] \mapsto [(P_b|\sigma_a)]$$

$$T_4 = [\underbrace{(P_a|(\sigma_b \wedge s))}_{T_0}] \mapsto [(P_a|\sigma_b)]$$

$$T_5 = [\underbrace{(P_b|(\sigma_b \wedge \Delta_2))}_{T_2}] \mapsto [(P_b|(\sigma_b)]$$

$$T_6 = [\underbrace{(P_a|(\sigma_a \wedge \Delta_3))}_{T_1}] \mapsto [(P_a|(\sigma_a)]$$

$$T_7 = [\underbrace{(\rho_a|(\sigma_a \wedge \sigma_b \wedge \Delta_4))}_{T_0}] \mapsto [(\rho_a|\sigma_b)]$$

$$T_8 = [\underbrace{(\rho_a|(\sigma_a \wedge \sigma_b \wedge \Delta_4))}_{T_0}] \mapsto [(\rho_a|\sigma_a)]$$

$$S = \{T_0, T_1, T_2, T_3, T_4, T_8\}$$

[Everything goes as per plan and Alice gets back her premium]

$$F_1 = \{T_0, T_1, T_6, T_8\}$$

[Bob aborts before committing P_b. Alice gets no compensation]

$$F_2 = \{T_0, T_1, T_2, T_5, T_6, T_7\}$$

[Alice aborts before redeeming P_b. Bob gets ρ_a as compensation]

Figure 2: Han-Lin-Yu Atomic Swap with 1 Premium

XH-swap: XH-Swap, as shown in Figure 3, requires both Alice and Bob to deposit premiums upfront: ρ_a, ρ_b, with $\rho_a > \rho_b$. This inequality ensures that in certain failure scenarios, if the smaller ρ_b goes to Alice and the larger ρ_a goes to Bob, Bob is effectively getting the premium $\rho_a - \rho_b$. If the principal amounts are equal ($P_a = P_b$), then Alice's premium is double that of Bob so that $\rho_a - \rho_b = \rho_b$. These premiums are committed to the blockchain upfront, with future-looking conditions (using the opcode `OP_LOOKUP_OUTPUT`) that govern whether these premiums go to Alice or Bob, depending on whether they take part in the swap, or abort the swap. The next set of transactions are equivalent to the TN-swap, but with additional conditions on where the premiums go. Due to the premiums being timelocked, two additional failure scenarios (on top of the two original failure scenarios of the TN-swap) have to be handled by XH-swap: parties aborting after their premiums are committed but before their principals are committed. Together, these four failure scenarios are shown in Figure 3, where in each scenario, either Alice or Bob aborts, either after committing their premiums or principals. The griefing costs of XH-swap are:

$S_{all} = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}\}$

$T_0 = [(\rho_a|\sigma_a)] \mapsto [(\rho_a|((\sigma_a \wedge \sigma_b \wedge \Delta_5) \wedge ((\overline{T_3.\Delta_2} \wedge (\overline{T_4.\Delta_2} \vee \overline{T_6.\Delta_5})) \vee \neg\overline{T_3.\Delta_2}))]$

$T_1 = [(\rho_b|\sigma_b)] \mapsto [(\rho_b|((\sigma_a \wedge \sigma_b \wedge \Delta_6) \wedge ((\overline{T_2.\Delta_1} \wedge (\overline{T_5.\Delta_3} \vee \overline{T_7.\Delta_6})) \vee \neg\overline{T_2.\Delta_1}))]$

$T_2 = [(P_a|\sigma_a)] \mapsto [(P_a|(\sigma_a \wedge \Delta_4) \vee (\sigma_b \wedge H_s))]$

$T_3 = [(P_b|\sigma_b)] \mapsto [(P_b|(\sigma_a \wedge H_s) \vee (\sigma_b \wedge \Delta_3))]$

$T_4 = \underbrace{[(P_b|(\sigma_a \wedge s))]}_{T_3} \mapsto [(P_b|\sigma_a)]$

$T_5 = \underbrace{[(P_a|(\sigma_b \wedge s))]}_{T_2} \mapsto [(P_a|\sigma_b)]$

$T_6 = \underbrace{[(P_b|(\sigma_b \wedge \Delta_3))]}_{T_3} \mapsto [(P_b|(\sigma_b)]$

$T_7 = \underbrace{[(P_a|(\sigma_a \wedge \Delta_4))]}_{T_2} \mapsto [(P_a|(\sigma_a)]$

$T_8 = \underbrace{[(\rho_a|(\sigma_a \wedge \sigma_b \wedge \Delta_5))]}_{T_0} \mapsto [(\rho_a|\sigma_a)]$

$T_9 = \underbrace{[(\rho_a|(\sigma_a \wedge \sigma_b \wedge \Delta_5))]}_{T_0} \mapsto [(\rho_a|\sigma_b)]$

$T_{10} = \underbrace{[(\rho_b|(\sigma_a \wedge \sigma_b \wedge \Delta_6))]}_{T_1} \mapsto [(\rho_b|\sigma_a)]$

$T_{11} = \underbrace{[(\rho_b|(\sigma_a \wedge \sigma_b \wedge \Delta_6))]}_{T_1} \mapsto [(\rho_b|\sigma_b)]$

$S = \{T_0, T_1, T_2, T_3, T_4, T_5, T_8, T_{11}\}$

[Everything goes as per plan and Alice gets back her premium]

$F_1 = \{T_0, T_8\}$

[Bob aborts before committing ρ_b. Alice gets no compensation]

$F_2 = \{T_0, T_1, T_8, T_{11}\}$

[Alice aborts before committing P_a. Bob gets no compensation]

$F_3 = \{T_0, T_1, T_2, T_7, T_8, T_{10}\}$

[Bob aborts before committing P_b. Alice gets ρ_b compensation]

$F_4 = \{T_0, T_1, T_2, T_3, T_6, T_7, T_9, T_{10}\}$

[Alice aborts before redeeming P_b. Bob gets $(\rho_a - \rho_b)$ as compensation]

Figure 3: Xue-Herlihy Atomic Swap with 2 Premiums

$$cost_{F_1} = f(\rho_a \cdot \Delta_5) > 0 \quad (7)$$

$$cost_{F_2} = f(\rho_b \cdot \Delta_6) > 0 \quad (8)$$

$$cost_{F_3} = f(\rho_a \cdot \Delta_5) + f(P_a \cdot \Delta_4) - \rho_b = 0 \quad (9)$$

$$cost_{F_4} = f(\rho_b \cdot \Delta_6) + f(P_b \cdot \Delta_3) - (\rho_a - \rho_b) = 0 \quad (10)$$

As XH-swap is explicitly designed to handle failure scenarios $\texttt{XH-F}_3$ and $\texttt{XH-F}_4$, the griefing costs C_{F_3} and C_{F_4} are 0 in Equations 9 and 10. However, XH-swap does not compensate Alice and Bob for their premiums. In case their counterparty aborts during the premium setup phase ($\texttt{XH-F}_1$ for Alice, or $\texttt{XH-F}_2$ for Bob) Alice and Bob receive no compensation. These are shown in Equations 7 and 8. To get these griefing costs close to zero, the authors of XH-swap propose using smaller premiums to bootstrap larger premiums, till the premiums are sufficient enough to swap the principals. The first set of premiums in such a *premium-chain* can be griefed, as it's backed by nothing. It is assumed that these premiums are small enough for Alice and Bob to ignore the griefing cost.

C. Shortcomings of Atomic Swaps with Premiums

There are four shortcomings in these protocols with premiums:

1) They do not compensate for locked-up premiums.
2) They are not compatible with Bitcoin.
3) Their final timelock is much longer than the classic TN-swap.
4) The the number of transactions under all scenarios (sizes of S_{all}, S, and F_i) are higher than the classic TN-swap.

These four shortcomings can all be attributed to a more fundamental idea that is embedded in these protocols – which is that of separating the *premium protocol* from the *principal protocol*. The latter is the classic TN-swap, and the former is bolted on the TN-swap to make it partially grief-free. As we see next, if we couple the two protocols together, we can overcome all four shortcomings.

IV. Grief-free Atomic Swap

Our Grief-free Atomic Swap (GF-swap) protocol's transactions are shown in Figure 4 and the actual flow of transactions between Alice and Bob are shown in the flowchart in Figure 5. As said before, the key insight that makes the protocol grief-free is the coupling between the premium and the principal protocols. The coupling is accomplished in two separate points.

1) A party's principal-committing transaction also commits to the counterparty's premium.
2) The same secret preimage is used to lock principals and the premiums in their hashlock arms.

Before we look at the swap in greater detail, a word about the premiums. As with the XH-swap, the GF-swap relies on the inequality $\rho_a > \rho_b$, given Alice and Bob's premiums ρ_a, ρ_b. The compensations that Alice and Bob get, in case they incur grief, are ρ_b and $\rho_a - \rho_b$ respectively. If the atomic swap is happening across different blockchains, say Bitcoin and Litecoin, Alice's principal P_a and Bob's premium ρ_b are on Bitcoin while Bob's principal P_b and Alice's premium ρ_a are on Litecoin. If the swap is happening on the same blockchain, both principals and both premiums are on that blockchain.

$S_{all} = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$

$T_0 = [(\rho_a|\sigma_a)] \mapsto [(\rho_a|(\sigma_a \wedge \sigma_b) \vee (\sigma_a \wedge H_s))]$

$T_1 = [(P_a|\sigma_a), (\rho_b|\sigma_b)] \mapsto [((P_a + \rho_b)| (\sigma_b \wedge H_s) \vee (\sigma_a \wedge \Delta_2))]$

$T_2 = [(P_b|\sigma_b), \underbrace{(\rho_a|(\sigma_a \wedge \sigma_b))}_{T_0}] \mapsto [((P_b + \rho_a)| ((\sigma_a \wedge H_s) \vee (\sigma_b \wedge \Delta_1))]$

$T_3 = [\underbrace{((P_b + \rho_a)|(\sigma_a \wedge s))}_{T_2}] \mapsto [((P_b + \rho_a)|\sigma_a)]$

$T_4 = [\underbrace{((P_a + \rho_b)|(\sigma_b \wedge s))}_{T_1}] \mapsto [((P_a + \rho_b)|\sigma_b)]$

$T_5 = [\underbrace{(\rho_a|(\sigma_a \wedge s))}_{T_0}] \mapsto [(\rho_a|(\sigma_a)]$

$T_6 = [\underbrace{((P_a + \rho_b)|(\sigma_a \wedge \Delta_2))}_{T_1}] \mapsto [((P_a + \rho_b)|(\sigma_a)]$

$T_7 = [\underbrace{((P_b + \rho_a)|(\sigma_b \wedge \Delta_1))}_{T_2}] \mapsto [((P_b + \rho_a)|(\sigma_b)]$

$S = \{T_0, T_1, T_2, T_3, T_4\}$

$F_1 = \{T_0, T_5\}$

[Bob aborts before committing ρ_b. Alice loses nothing.]

$F_2 = \{T_0, T_5\}$

[Alice aborts before committing P_a. Bob loses nothing.]

$F_3 = \{T_0, T_1, T_6, T_5\}$

[Bob aborts before committing P_b. Alice gets ρ_b as compensation]

$F_4 = \{T_0, T_1, T_2, T_6, T_7\}$

[Alice aborts before redeeming P_b. Bob gets $(\rho_a - \rho_b)$ as compensation]

Figure 4: Grief-Free Atomic Swap with 2 Premiums

A. Setup

Refer to Figure 5 for the following numbered steps.

1) Alice creates GF−T$_0$, which locks her premium ρ_a such that it can be unlocked either by a multisig signed by both Alice and Bob, or just by Alice if she also reveals the secret preimage s of hash H_s. Note that GF−T$_0$ has no timelock. Alice sends GF−T$_0$ to Bob so that he can inspect it. GF−T$_0$ is not broadcast to the blockchain yet.
2) Bob hands over his premium ρ_b to Alice off-chain. This premium is a UTXO that Bob controls. Bob can also prove that he can spend this UTXO by signing a standard "Hello World" message with the public key that locks ρ_b. Note that such a signature just confirms to Alice that Bob controls ρ_b, and she cannot do anything else with such a signature.
3) Alice constructs GF−T$_1$ which commits Alice's principal P_a. This transaction will also include include a reference to ρ_b. Alice sends GF−T$_1$ to Bob. Before signing GF−T$_1$,

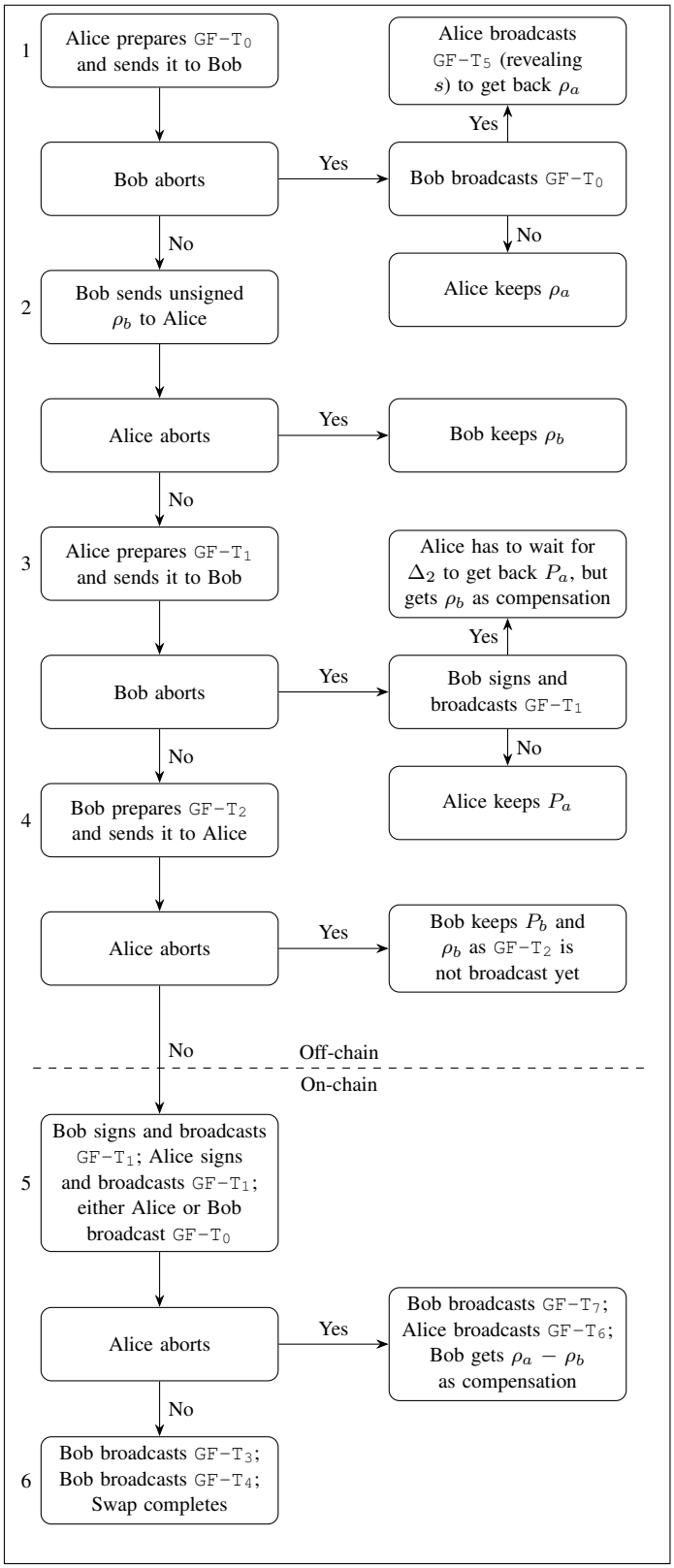

Figure 5: Grief-free Atomic Swap - Transaction Flow

Bob ensures that it pays Alice's premium to him if he reveals the secret preimage of the already known hash from GF−T$_0$. Note that Bob has already seen GF−T$_0$ and can match the H_s from GF−T$_0$ and GF−T$_1$. GF−T$_1$ also has a refund arm going back to Alice, which has a timelock.

4) Bob then constructs his principal committing transaction GF−T$_2$ which also uses Alice's premium ρ_a. To do this, Alice has to give her signature to Bob so that the multisig that locks ρ_a can be unlocked in GF−T$_2$. Alice does this only if GF−T$_2$ sends both Bob's principal and Alice's premium ($P_b + \rho_a$) to Alice, if she reveals the preimage of the same hash H_s. GF−T$_2$ also has a refund arm going back to Bob, which has a timelock.

The series of transactions GF−T$_0$, GF−T$_1$, and GF−T$_2$ can be constructed and signed off-chain in the specific order mentioned above, and broadcasted by either party in Step 5. Both GF−T$_1$ and GF−T$_2$ take two inputs each, a party's principal and the counterparty's premium, and send their sum to the redeeming party if they reveal the secret preimage, or refund it back to the party if they wait for timelocks to expire - just like in TN-swap. The principals and the premiums are coupled now. After the setup stage, we look at how the rest of the swap can play out, including success and failure scenarios.

B. Success

If the swap goes as per plan and we reach Step 6, Alice broadcasts GF−T$_3$ to redeem $P_b + \rho_a$ and Bob broadcasts GF−T$_4$ to redeem $P_a + \rho_b$. Both parties get their counterparty's principal and their own premiums back.

C. Failures

The protocol handles the four failure scenarios gracefully and grief-free. In the following failure scenarios, we look at only those cases where a party is being griefed due to their counterparty aborting. If a party aborts on their own volition and has to refund their principal amount back to themselves after a timelock, we do not consider it a failure scenario.

Bob aborts before commiting ρ_b (GF−F$_1$): During the off-chain interaction where GF−T$_0$, GF−T$_1$, and GF−T$_2$ are being constructed, Bob could abort and not give his signature to GF−T$_1$ even if Alice has constructed it properly. In this case, as nothing has been broadcast on the blockchain, there is no grief. On the other hand, after GF−T$_0$, GF−T$_1$, and GF−T$_2$ are constructed, broadcast, but not confirmed, Bob could double-spend ρ_b in a parallel transaction. In this case, Alice gets back her premium without delay using GF−T$_5$ by revealing the preimage. Revealing this preimage is harmless to Alice as her principal (which can be withdrawn by Bob if he knows this preimage) cannot be confirmed on the blockchain as Bob made GF−T$_1$ invalid by spending ρ_b elsewhere. If Bob is careless and makes only GF−T$_1$ invalid by spending ρ_b, and leaves GF−T$_2$ valid and allowed to confirm on the blockchain - he risks losing his principal P_b as well, as Alice can broadcast GF−T$_3$ and claim both the principals and her own premium. If Bob wants to abort at this stage in good faith, he has to not give his signatures to Alice for GF−T$_1$ and GF−T$_2$. Nothing hits the blockchain, and both parties lose nothing.

Alice aborts before committing P_a (GF−F$_2$): Alice's principal P_a and Bob's premium ρ_b are committed to the blockchain in a single transaction GF−T$_1$, and hence this scneario cannot occur. As in, Alice cannot abort and still grief Bob because if Alice aborts here, Bob's premium never hits the blockchain and there is no question of griefing Bob.

Bob aborts before committing P_b (GF−F$_3$): After constructing, signing, and broadcasting GF−T$_0$, GF−T$_1$, and GF−T$_2$, Bob could double spend P_b in a parallel transaction, thereby making GF−T$_2$ invalid. Alice can then get Bob's premium by confirming GF−T$_6$, and also get her own premium back with GF−T$_5$. Note that GF−T$_5$ is valid because Bob made the other transaction spending ρ_a (GF−T$_2$) invalid by double spending P_b elsewhere. Alice has to wait for the timelock of Δ_2, and for that, she is compensated with Bob's premium ρ_b.

Alice aborts before redeeming P_b (GF−F$_4$): After constructing, signing, and broadcasting GF−T$_0$, GF−T$_1$, and GF−T$_2$, it is Alice's turn to redeem P_b by broadcasting GF−T$_3$. If she doesn't do it while time Δ elapses, Bob claims his refund back with GF−T$_7$. Alice could claim her own refund back with GF−T$_6$. In this case, Bob gets Alice's premium ρ_a and Alice gets Bob's premium ρ_b. As $\rho_a > \rho_b$, it is Bob who is compensated here with the premium $\rho_a - \rho_b$ because Alice aborted the swap.

Setup Signatures: During the construction and signing of GF−T$_0$, GF−T$_1$, and GF−T$_2$, we have Bob signing for his premium in GF−T$_1$ and Alice signing her premium in GF−T$_2$ (created by GF−T$_0$'s multisig output arm). Bob has to make sure that Alice has signed GF−T$_2$ and given him a copy before he signs GF−T$_1$. This ordering handles two separate failure scenarios:

1) Bob waits for Alice to sign GF−T$_2$ and give him a copy before signing GF−T$_1$ himself and giving her a copy. So, we are now either in the scenarios GF−F$_1$, GF−F$_2$, or GF−S. Bob loses nothing in all of these.
2) Alice signs GF−T$_2$, but does not get Bob's signature on GF−T$_1$. Bob has two choices now.
   a) Bob can either keep GF−T$_2$ without broadcasting it, and we are in GF−F$_1$ where Alice doesn't lose anything.
   b) Bob can broadcast GF−T$_2$, and risk losing his principal P_b to Alice as well. To avoid that, he has to sign and broadcast GF−T$_1$ and we are in GF−F$_4$, or GF−S. Alice loses nothing in these.

Cost: The griefing costs of GF-swap are:

$$cost_{F_1} = 0 = 0 \quad (11)$$

$$cost_{F_2} = 0 = 0 \quad (12)$$

$$cost_{F_3} = f(P_a \cdot \Delta_2) - \rho_b = 0 \quad (13)$$

$$cost_{F_4} = f(P_b \cdot \Delta_1) + f(\rho_b \cdot \Delta_2) - (\rho_a - \rho_b) = 0 \quad (14)$$

As seen in failure scenarios $\texttt{GF-F}_1$ and $\texttt{GF-F}_2$, if parties abort before committing their principals, no timelocks are engaged, and we get $cost_{F_1} = 0$ and $cost_{F_2} = 0$ in Equations 11 and 12. Additionally, in failure scenario $\texttt{GF-F}_3$, $f(P_a \cdot \Delta_2)$ is compensated by ρ_b, and therefore $cost_{F_3} = 0$. Here, Alice's premium ρ_a is never committed, and ρ_b does not have to compensate for it. In failure scenario $\texttt{GF-F}_4$, both $f(\rho_b \cdot \Delta_1)$ and $f(P_b \cdot \Delta_1)$ together have to be compensated by $\rho_a - \rho_b$ to get $cost_{F_4} = 0$.

D. Coupling Principal and Premium

Section III-C listed the four shortcomings of previous premium-based designs. By coupling the *premium protocol* and the *principal protocol*, GF-swap manages to avoid these shortcomings.

Premium Lockup Compensation: Coupling the principal and the premium protocols lets us use the same values of the timelock for both. This ensures that wherever the principal goes, with whatever delay, the premium also follows. The only catch here is Alice's premium, which is locked in $\texttt{GF-T}_0$. But this specifically avoids a timelock and is hence grief-free.

Bitcoin Compatibility: Decoupling the two protocols forces the premium protocol to lookup where the principal was sent, which needs a new Bitcoin opcode `OP_LOOKUP_OUTPUT`. Coupling them sends the two: premium and principal, together to their destination, and we do not need `OP_LOOKUP_OUTPUT`. It can otherwise be argued that, from a software engineering perspective, decoupling the protocols is better than coupling them. But the moment we decouple the two protocols, there is no way to construct the premium protocol without knowing how the principal protocol will play out in the future. In our opinion, Bitcoin compatibility is as important as the software engineering decoupling.

Timelock Length: Coupling the protocols lets GF-swap keeps the timelock values of TN-swap, as the premiums themselves do not need separate timelocks of larger values. This reduces the time it takes to make a full swap, when compared to related work presented earlier.

Number of Transactions: Coupling let's us go just 1 transaction over TN-swap ($\texttt{GF-T}_0$, which sets up Alice's premium ρ_a). It's an open question whether we can incorporate premiums into a grief-free protocol while keeping the total number of transactions the same as TN-swap.

E. Disadvantages of Coupling:

As discussed before, we choose to couple the premium and principal protocols to achieve Bitcoin compatibility. As expected, this design "anti-pattern" makes it harder to extend GF-swap to handle other use cases. Examples: Bribing free atomic swaps [15] [16], Multi-party swaps across more than 2 blockchains [17], atomic swap enabled automated market makers. Out of these, the bribing-free atomic swap from [15] can be also made grief-free with minor tweaking of the transactions involved. Alice's timelocked refund arm from $\texttt{GF-T}_1$ has to be made to go through another layer to bring in the extra fees that Alice needs to commit to make the combined protocol bribe-free as well.

Payment channels and atomic swaps both use HTLCs as a building block. The GF-swap has a modified version of the HTLC. It is an open question whether this modified HTLC can be used to construct payment channels.

V. Conclusion

We have proposed an atomic swap protocol that makes the classic Tier Nolan swap resilient to griefing while adding just one extra on-chain transaction. We compensate griefing by offering a *premium* to the party that gets griefed. Most of the heavy-lifting in our swap is done off-chain, where the two parties communicate to establish the swap in the first place. Unlike other protocols, in our swap, both parties can abort the premium protocol off-chain and not on-chain. We also show that coupling the premium and the principal protocol makes the swap implementable in Bitcoin, where transaction execution does not have access to an external global state. The coupling also reduces transaction costs and the worst-case timelock. Unfortunately, this coupling makes the GF-swap non-trivial to extend to other applications without careful tweaking of transactions.

References

[1] R. Cleve, "Limits on the security of coin flips when half the processors are faulty," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, 1986, pp. 364–369.

[2] H. Pagnia and F. C. Gärtner, "On the impossibility of fair exchange without a trusted third party," Citeseer, Tech. Rep., 1999.

[3] M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997, pp. 1–5.

[4] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1998, pp. 591–606.

[5] "Atomic Swaps," https://bitcointalk.org/index.php?topic=193281.msg2224949, [Accessed: 2020-05-07].

[6] D. Robinson, "HTLCs Considered Harmful," https://cyber.stanford.edu/sites/g/files/sbiybj9936/f/htlcs_considered_harmful.pdf, [Accessed: 2020-05-07].

[7] G. Maxwell, "Zero knowledge contingent payment. 2011," *URl: https://en. bitcoin. it/wiki/Zero Knowledge Contingent Payment (visited on 05/01/2016)*, 2016.

[8] F. K. Maurer, "A survey on approaches to anonymity in bitcoin and other cryptocurrencies," *Informatik 2016*, 2016.

[9] E. Heilman, S. Lipmann, and S. Goldberg, "The arwen trading protocols," in *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, ser. Lecture Notes in Computer Science, J. Bonneau and N. Heninger, Eds., vol. 12059. Springer, 2020, pp. 156–173. [Online]. Available: https://doi.org/10.1007/978-3-030-51280-4_10

[10] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 967–984.

[11] L. Eckey, S. Faust, and B. Schlosser, "Optiswap: Fast optimistic fair exchange," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 543–557.

[12] R. Han, H. Lin, and J. Yu, "On the Optionality and Fairness of Atomic Swaps," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, ser. AFT '19. Association for Computing Machinery, pp. 62–75.

[13] Y. Xue and M. Herlihy, "Hedging against sore loser attacks in cross-chain transactions," ser. PODC'21, 2021.

[14] G. Avarikioti, O. S. T. Litos, and R. Wattenhofer, "Cerberus channels: Incentivizing watchtowers for bitcoin," Cryptology ePrint Archive, Report 2019/1092, 2019, https://ia.cr/2019/1092.

[15] T. Nadahalli, M. Khabbazian, and R. Wattenhofer, "Timelocked Bribing," in *Financial Cryptography and Data Security (FC), Online*, March 2021.

[16] I. Tsabary, M. Yechieli, and I. Eyal, "MAD-HTLC: Because HTLC is Crazy-Cheap to Attack," 2020.

[17] M. Herlihy, "Atomic Cross-Chain Swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, pp. 245–254.