

"במהירות הבזק"

כיצד מתבצעות טרנזקציות ברשת הברק

הסבר ממוקד על איך [המימוש](#) של רשת הברק עובד. נדרש ידע קודם של: [UTXO, Bitcoin Script, and digital signatures](#).

תרגום מאנגלית של המאמרים:

<https://medium.com/swlh/till-its-lightning-fast-uncover-the-development-of-the-lightning-network-fbdo1bcoe8o>

<https://medium.com/@yyforyongyu/till-its-lightning-fast-uncover-the-lightning-network-transactions-f318oe467857>

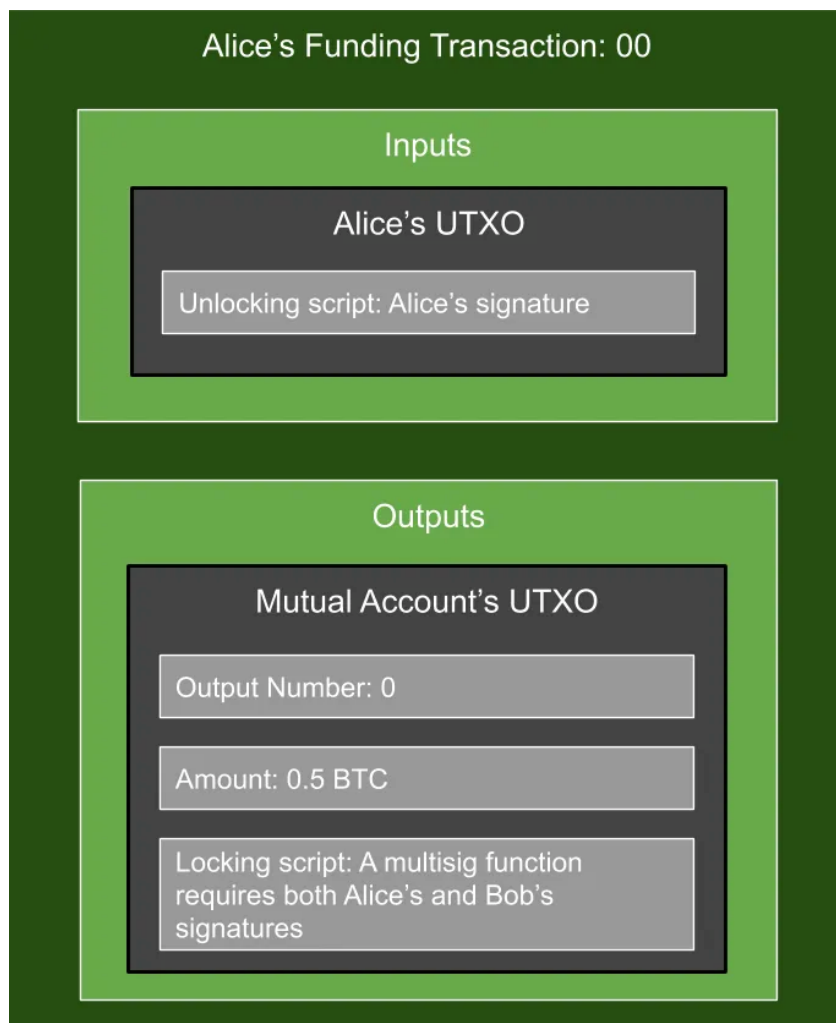
פרק א' – פיתוח רשת הברק

הרעיון של רשת הברק הוא ליצור מערכת שתפעל לצד רשת הביטקוין. בעוד שהטרנזקציות בביטקוין חייבות להיות מוגבלות בזמן ובגודל, כלומר שב-2MB (לערך) יישלחו כל 10 דקות, רשת הברק פורצת את המגבלות הללו ע"י העברת ביטקוינים לרשת שלה וסירקולציה שלהם בתוך הרשת תחת מגבלות אחרות.

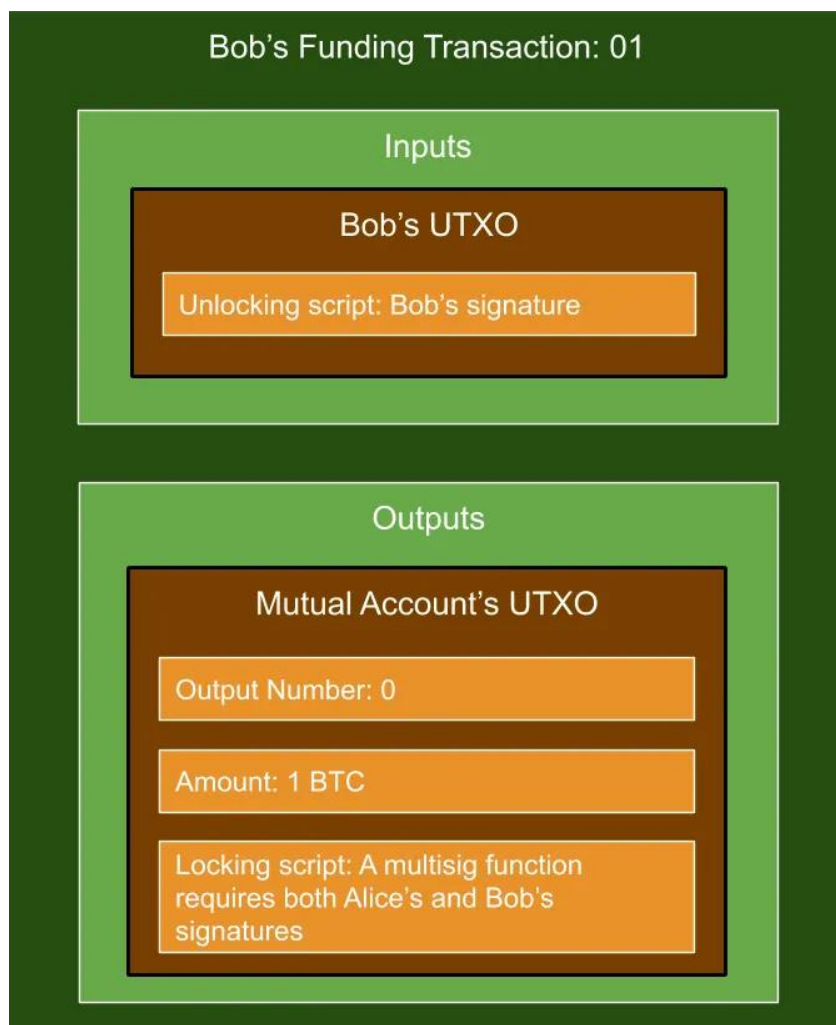
אם שני משתתפים מעוניינים להצטרף לרשת הברק, הצעד הראשון עבורם הוא ליצור כתובת מרובת-חתימות (multisig) עבור חשבון משותף, או במילים אחרות לפתוח ערוץ בינן. מהרגע שנפתח הערוץ, הוא ינוצל להעברת כסף בתוך הרשת. אם מאיזושהי סיבה צריך למשוך את הכסף מהערוץ לרשת הביטקוין, המשתתפים יכולים לסגור את הערוץ בשיתוף פעולה. הכסף זורם מרשת הביטקוין לרשת הברק אם אנשים מחליטים לפתוח ערוצים, והוא אף פעם לא צריך לחזור לרשת הביטקוין אלא אם כן אנשים רוצים לסגור את הערוצים שלהם. באופן כללי, שני המקרים הצפויים לטרנזקציות "בין שכבות" הן פתיחת ערוץ וסגירת ערוץ, פרט למקרה יוצא הדופן: כשאחת המשתתפים הפסיקה להגיב או מנסה לפעול באופן זדוני – שאר המשתתפים יכולים באופן חד צדדי, ללא שיתוף פעולה איתו, לסגור את הערוצים מולה.

בניית חשבון משותף

כדי להשתמש ברשת הברק, הצעד הראשון הוא להעביר אליה כמה ביטקוין. המשתתפים ייצרו כתובות מרובות-חתימות שאליהן ישלחו את הכספים. לדוגמא, אליס ובוב מחליטים לפתוח ערוץ עם 1.5 ביטקוין, שאליו אליס תתרום 0.5 ביטקוין ובוב יתרום 1 ביטקוין. כל אחת מהם תיצור טרנזקציה כדי לתקצב את הערוץ, ואת הכסף שנעול בכתובת הזו לא ניתן להוציא אלא אם כן שניהם מסכימים.



ה-ID של הטרנזקציה וה-Output Number של הכניסות
הושמטו לצורך ההדגמה.



ה-ID של הטראנזקציה וה-Output Number של הכניסות
הושמטו לצורך ההדגמה.

נתייחס עכשיו למקרים הבאים:

- מה קורה אם אחת המשתתפות מסרבת לשתף פעולה?
- מה קורה אם אליס, בוב, או שציהקן ביחד, מחליטים לסגור את הערוץ ולקבל את כספן בחזרה?
- מה אם צריך לעדכן את המאזן? נניח אליס משלמת לבוב 0.1 ביטקוין עבור משהו שהיא קונה מבוב?

לפני שנתמודד עם הנושאים הללו, חשוב להדגיש שרשת הברק צריכה להיות מתוכננת באופן המאפשר אבטחה תוך שמירה על ביזור. היא צריכה להישאר Trustless כמו רשת הביטקוין - ולא לסמוך על היושרה של אף אחת או אף ארגון כדי להעביר כסף בהצלחה. לשם כך, נדרשת תכנית מורכבת באמצעות ה-Bitcoin Script. בניגוד לרשת הביטקוין המסתמכת על כוח חישוב, רשת הברק תאבטח את עצמה באמצעות עונשים מובנים בתוך המערכת.

בואו נפתור תחילה את בעיית התקצוב הראשוני של הערוץ. נזכור שבשכבה הראשונה, רשת הביטקוין, טרנזקציית ביטקוין מתבצעת בשלושה שלבים:

1. יצירת הטרנזקציה

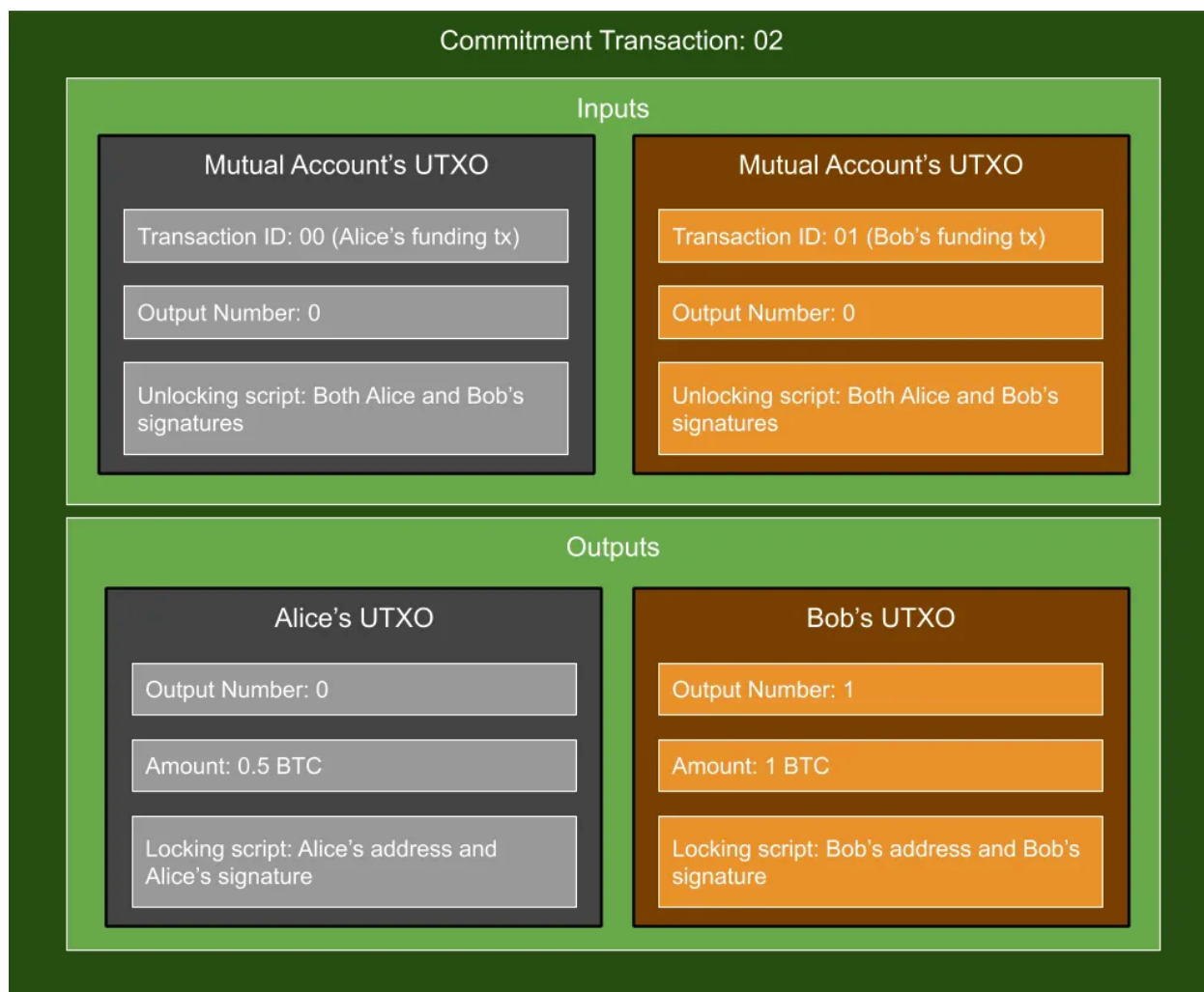
2. חתימה על הטרנזקציה - או בניסוח טכני: צירוף "סקריפט שחרור נעילה" (Unlocking script).

3. שידור הטרנזקציה

האם אנחנו יכולים לפתח מנגנון חכם בין השלבים הללו?

תקצוב ללא חתימה

אליס ובוב יכולים ליצור את הטרנזקציות לעיל בלי לחתום עליהן. המטרה של השארית הטרנזקציות לא חתומות היא כדי לאפשר את ביטול התהליך במקרה שאחד הצדדים מסרב לשתף פעולה. מכיוון שהטרנזקציות לא חתומות, אף אחת לא מחויבת לשום דבר, כלומר שום נזק לא יכול להיגרם באותו הרגע. באמצעות התייחסות ל-Outputים של הטרנזקציות הלא חתומות, אליס ובוב יכולים ליצור מראש טרנזקציה חדשה כדי להחליט לאן יגיעו הכספים שינעלו בכתובת מרובת-החתימות (multisig). במצב הנוכחי, הטרנזקציה הזו צריכה לתת 0.5 ביטקוין לאליס, ו-1 ביטקוין לבוב.



מהרגע שהטרנזקציה הזו נוצרה, שתי שאלות צריכות להישאל:

1. האם הן צריכות לחתום על הטרנזקציה החדשה? כן. ברגע זה אף אחד לא מסכנת שום כסף. אם אלס חתמה על הטרנזקציה אבל בוב מסרב, היא יכולה מייד לעצור את התהליך בלי לאבד שום דבר, ולהיפך.

2. האם הן צריכות לשדר את הטרנזקציה החדשה? לא. המטרה של הטרנזקציה הזו היא למנוע איבוד כספים במקרה שאחד המשתתפים לא מגיב. כל עוד הצד השני מגיב, וכל עוד המשתתפים לא מעוניינים לסגור את הערוץ, הטרנזקציה הזו לעולם לא תשודר.

מה שצריכה להיחתם ולהתפרסם במקום זה, זוהי טרנזקציית התקצוב המקורית, ברגע שלאליס ולבוב יש עותק חתום של הטרנזקציה החדשה. הטרנזקציה החדשה מתפקדת בדומה לעירבון במקרה של שכירת דירה – נותנים אותה כדי להראות מחויבות לחוזה, והיא מאפשרת לקבל פיצוי במקרה שהחוזה הופר. לכן לטרנזקציה זו קוראים בדר"כ בשם "טרנזקציית ההתחייבות" (Commitment Transaction).

לסיכום, התהליך שבו אליס ובוב פותחים ערוץ באופן מאובטח הוא כדלקמן:

1. אליס ובוב, כל אחד בנפרד, יוצרים טרנזקציית תקצוב מבלי לחתום עליה.

2. אליס ובוב יוצרים וחתימתם על טרנזקציית התחייבות המבוססת על טרנזקציות התקצוב. לדוגמא: אליס יוצרת את טרנזקציית ההתחייבות, חותמת עליה, שולחת לבוב, ובוב בודק אותה ושולח את חתימתו חזרה לאליס. או ששניהם בונים את טרנזקציית ההתחייבות בנפרד ושולחים אחת לשניה רק את החתימות. בכל מקרה, התוצאה היא ששניהם מחזיקים באותה טרנזקציית התחייבות חתומה.

3. אליס ובוב משדרים את טרנזקציות התקצוב שלהם. ברגע שטרנזקציות התקצוב עברו confirmation, הערוץ נחשב "ממומן" והם צריכים למצוא דרך לשלוח דרכו ביטקוין. אבל לפני שאנחנו משחקים עם ה-Bitcoin Script כדי לאפשר עדכוני מאזן בערוץ, עלינו לשאול שאלה קריטיות: איך יוצרים בפועל את טרנזקציית התקצוב? הערת מתרגם: גם המקרה שבו שניהם מחזיקים בטרנזקציית ההתחייבות הנ"ל חתומה, אבל רק צד אחד משדר את טרנזקציית התקצוב שלו, הוא בעייתי - כי טרנזקציית ההתחייבות מצביעה על שני Input, ושניהם צריכים להיות ברשת הביטקוין כדי שהיא תתקבל. גם בעיה זו נפתרת ע"י תקצוב חד צדדי.

האלטרנטיבה - תקצוב חד צדדי

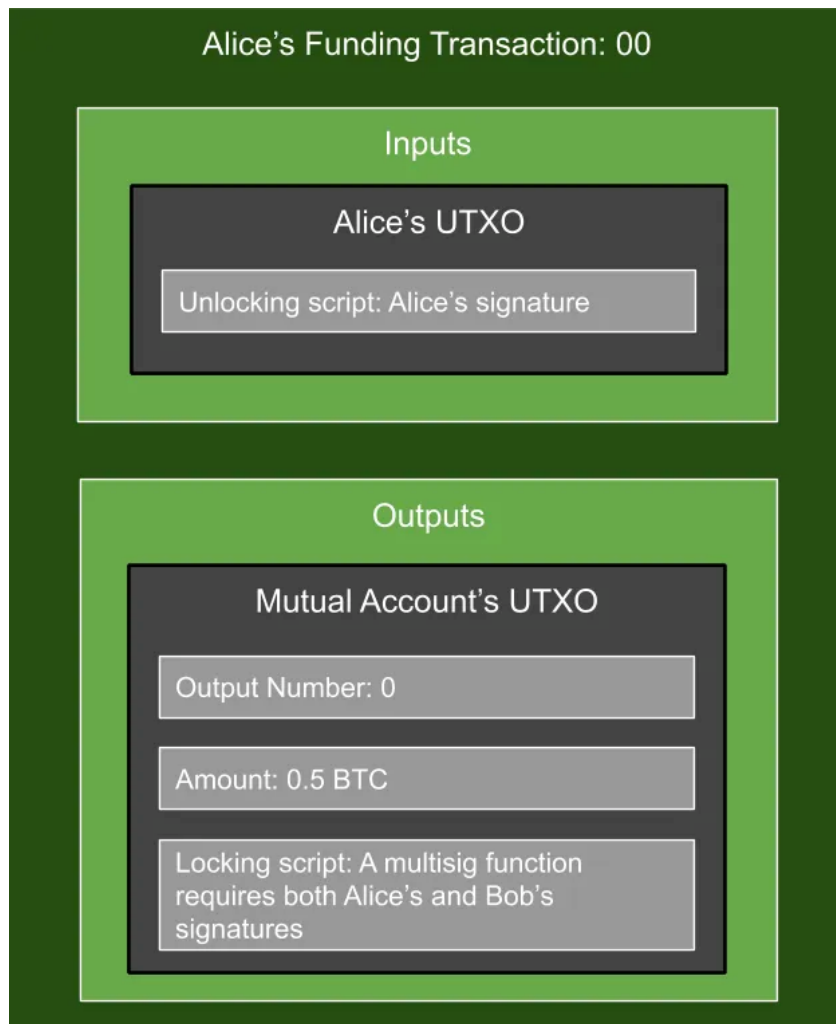
במקום תקצוב דו צדדי כמו שהוצג קודם, המימוש בפועל של רשת הברק משתמש בתקצוב חד צדדי עם טוויסט לגבי האופן שבו טרנזקציית התקצוב נחתמת ומשודרת. בשלב הראשון, כל משתתף יכול להיות ה-"מתקבצ". אם למשל אליס מחליטה לקנות משהו מבוב, היא יכולה להתייחס לתשלום כאל התקציב שבווב אמור לשים בחלק שלו בפתיחת הערוץ. בהמשך, כשבוב יבצע קניות מאחריה, הוא יוכל להשתמש בכסף בערוץ בלי שום אינטראקציה עם רשת הביטקוין (במימוש הסופי). ככל שמספר הערוצים עולה, לכל משתתף יש איזשהו מאזן באיזשהו ערוץ שפתחו מולה, באופן דומה למצב שבו הקמת הערוצים הייתה מתבצעת באופן דו צדדי. אם אליס הייתה המתקבצת הראשונית בתקצוב חד צדדי, זה קריטי לוודא שאם בוב מפסיק להגיב, אליס עדיין תוכל לקבל חזרה את התקצוב שלה.

כשבונים טרנזקציית ביטקוין, ה-Input שלה צריכים להפנות ל-ID של טרנזקציות קודמות ול-Output Number שלהן. לפיכך, אליס יכולה לבנות מראש את כל טרנזקציית התקצוב מבלי לשדר אותה, ולשתף עם בוב רק את ה-ID וה-Output Number. כשבוב מקבל את המידע, מכיוון שהוא לא קיבל מאליס את החתימות הדרושות לטרנזקציה, הוא לא יכול לעשות שום דבר עם המידע חוץ מאשר ליצור טרנזקציית התחייבות, שתיתן לאליס חזרה את הכסף. סיכום הצעדים הוא כדלקמן:

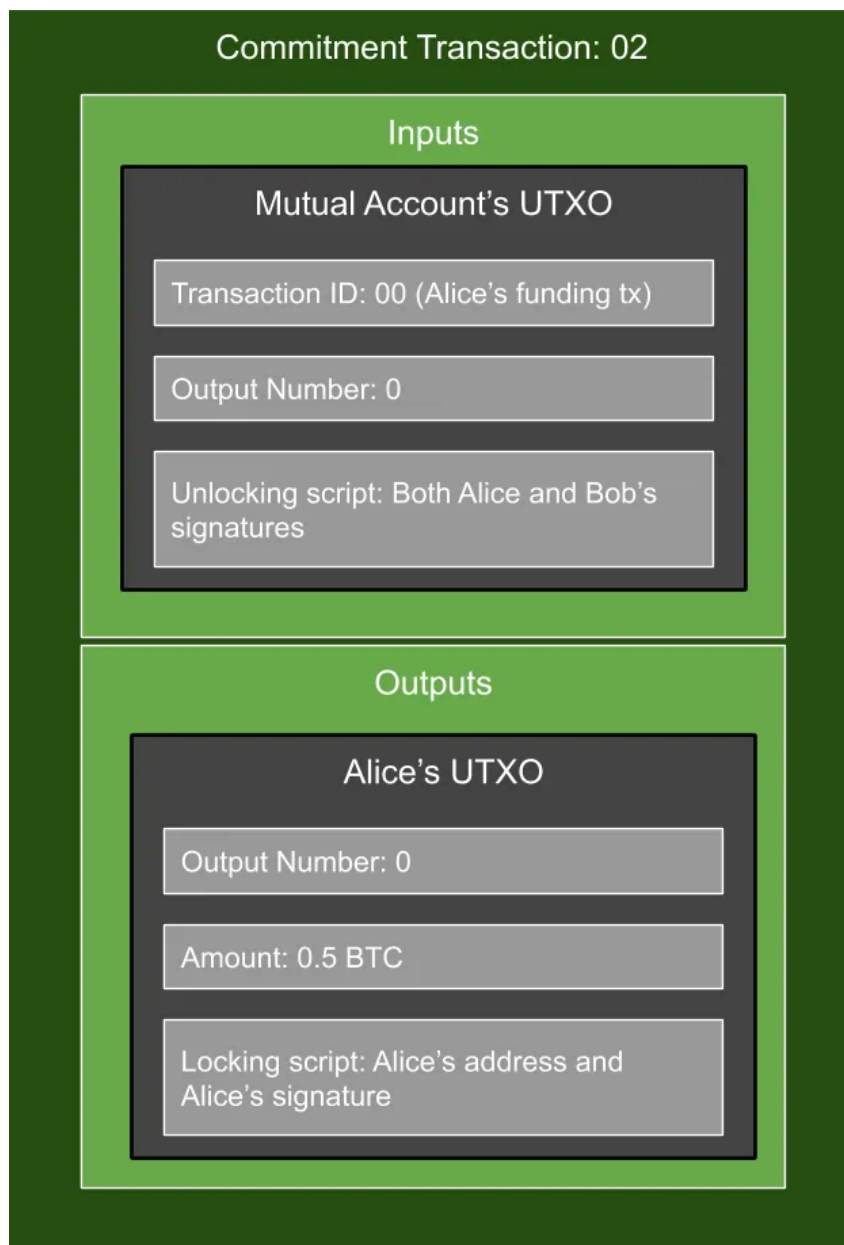
1. אליס יוצרת את טרנזקציית התקצוב, חותמת עליה, ומשתפת עם בוב את ה-ID וה-Output Number של הטרנזקציה (בלי הטרנזקציה עצמה ובלי החתימות שלה).
2. אליס ובוב יוצרים וחתימתם את טרנזקציית ההתחייבות, על בסיס ה-ID וה-Output Number של טרנזקציית התקצוב. שימו לב שטרנזקציית התקצוב שמה את הכסף של אליס בכתובת

מרבית-חתימות (multisig), שדורשת את החתימות של שניהם כדי לשלוח את הכסף חזרה לאליס באמצעות טרנזקציית ההתחייבות.

3. אליס משדרת את טרנזקציית התקצוב. בנקודה זו, אם בוב מפסיק להגיב, אליס יכולה לשדר את טרנזקציית ההתחייבות כדי לקבל את כספה חזרה.



טרנזאקציית התקצוב של אליס נשארת ללא שינוי, אבל היא לא תשדר אותה ולא תשתף עם בוב את חתימתה, אלא רק תשלח לו את ה-ID של הטרנזקציה (00) ואת ה-Output Number של הטרנזקציה (0).



שיטת התקצוב החד צדדי הזו מסירה את הסיכון שיאבדו כספים למתקצב, תודות לאופן שבו ביטקוין מבזבז UTXOים בטרנזקציות שלו. למרות שאפשר להרחיב את השיטה לתקצוב דו צדדי, [פרוטוקול רשת הברק בפועל](#) מעדיף את שיטת התקצוב החד צדדי. הסיבה לכך היא כנראה מכיוון שדרושה פחות תקשורת בין המשתתפים, למרות שזה לא מוסבר באופן מפורש. לגבי תקצוב דו צדדי, ישנו [דיון מתמשך](#) לגבי איך ניתן לממש אותו. הערת מתרגם: נכון לכתיבת מילים אלו יש מימוש שאולי תומך ב-dual-funding כזה, בשם [nolooking](#) (וגם [כאפליקציה ל-Umbrel](#)), אבל לא ניסיתי אותו בעצמי.

עדכון המאזן בערוץ

כעת יש לנו חשבון כספי משותף שניתן לסגור ולסיים, אבל מה לגבי שינוי המאזן בחשבון? אליס שמה בחשבון 0.5 ביטקוין לפתיחת הערוץ, והיא רוצה לשלוח 0.1 ביטקוין לבוב בתמורה ללפטופ שלו - איך הק יעשו זאת ברשת הברק?

הפתרון הטריטוריאלי הוא שאליס ובוב יכולים ליצור טרנזקציות חדשות כדי לשקף את המאזן האחרון בינם.

מבחינת אליס, היא תיצור את הטרנזקציה הבאה, תחתום עליה ותשלח אותה לבוב. מכיוון שהטרנזקציה חתומה כבר ע"י אליס, בוב יכול להוסיף את החתימה שלו ולשדר את הטרנזקציה מתי שירצה.

Alice's Transaction: 03

Inputs

Mutual Account's UTXO

Transaction ID: 00 (Alice's funding tx)

Output Number: 0

Unlocking script: Both Alice and Bob's signatures

Outputs

Alice's UTXO

Output Number: 0

Amount: 0.4 BTC

Locking script: Alice's address and Alice's signature

Bob's UTXO

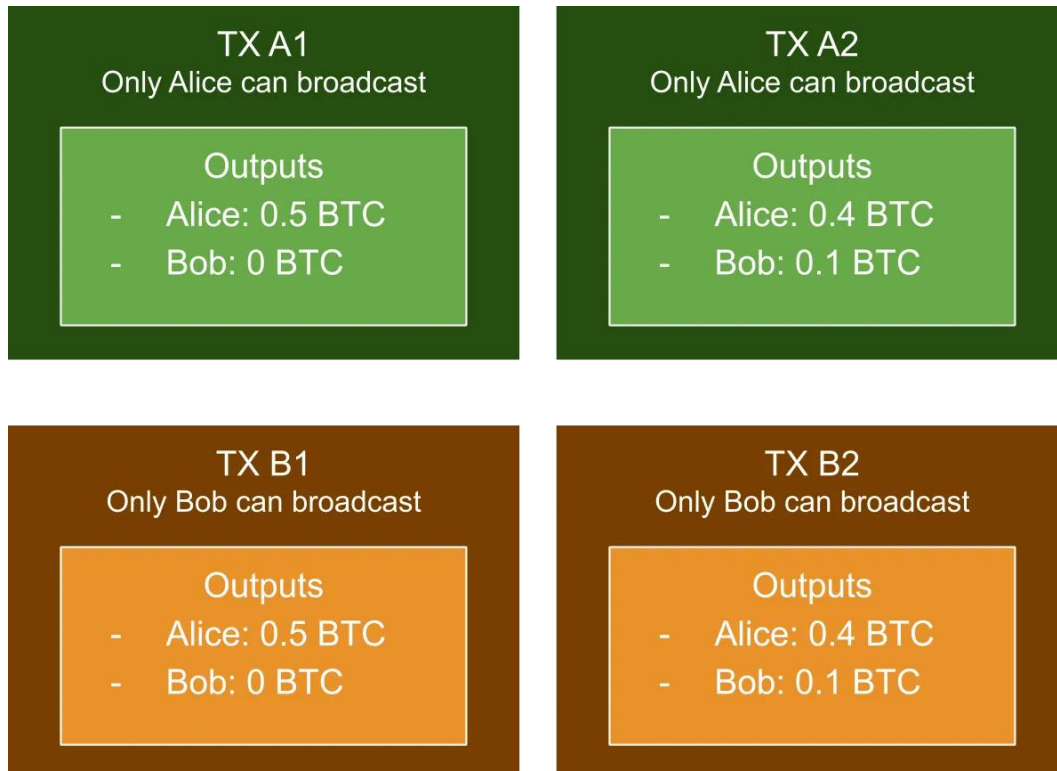
Output Number: 1

Amount: 0.1 BTC

Locking script: Bob's address and Bob's signature

באופן סימטרי, בוב יחתום על הטרנזקציה וישלח אותה לאליס, כדי שהיא תוכל לחתום ולשדר אם היא רוצה (אם היא רוצה לסגור את הערוץ מגיע לה לקבל רק 0.4 ביטקוין). כרגע, לשני המשתתפים יש שתי טרנזקציות חתומות במקביל. יחד, ראינו 5 טרנזקציות עד כה:

- טרנזקציית התקצוב מאליס, שבבר שודרה.
 - טרנזקציית ההתחייבות שאליס מחזיקה, שנותנת לה את הכסף חזרה. נקרא לה: TX A1.
 - בוב גם מחזיק בעותק של טרנזקציית ההתחייבות. נקרא לו: TX B1.
 - הטרנזקציה החדשה של אליס, שנותנת לאליס 0.4 ביטקוין, ולבוב 0.1 ביטקוין. נקרא לה: TX A2.
 - לבוב גם יש עותק של הטרנזקציה החדשה. נקרא לו: TX B2.
- הטרנזקציות שהמשתתפים מחזיקים ויכולים לשדר, בפשטות נראות ככה:



כל ארבע הטרנזקציות הן תקינות לפני ששודרו לרשת הביטקוין, ויש להן סיכוי שווה להתקבל ע"י כורים אם משדרים אותן באותו רגע. אם זה נשמע לכם מבלבל, דמיינו את ביטקוין בתור "הדוד תום" שעובד כמנהל חשבונות ומעדכן ספר חשבונות על פי הטרנזקציה הראשונה שהוא שומע, ומתעלם מהטרנזקציות הבאות.

במקרה הנוכחי, אם נתייחס לאליס ובוב כאל אנשים רציונליים (מבחינה כלכלית), האינטרס של אליס הוא לשדר את TX A1 כדי לגנוב את ה-0.1 ביטקוין שהיא הבטיחה לשלם לבוב. מבחינת בוב, עדיף לו לשדר את TX B2 מכיוון שהוא יקבל 0.1 ביטקוין (עדיף על כלום). אם שתי הטרנזקציות משודרות באותו זמן, מדובר רק בסיכויים של 50:50 שבו יקבל את מה שמגיע לו – וזה בלתי מתקבל על הדעת. איך נפתור את הבעיה? איך אנחנו יכולים לוודא שכל המשתתפים תמיד משדרים את הטרנזקציה האחרונה ולא טרנזקציות ישנות?

לפני שנתקדם הלאה, ניזכר בסוגי הטרנזקציות פה, מכיוון שזה עלול להיות מבלבל כשמסתכלים על כל הטרנזקציות שהתרחשו ברשת הברק.

- אם לטרנזקציה אין שום קשר לרשת הברק, כלומר היא מתרחשת רק ברשת הביטקוין, קוראים לה טרנזקציית "שכבה ראשונה".
- אם הטרנזקציה מכניסה כסף מרשת הביטקוין לרשת הברק, כלומר פותחת ערוץ, או שהיא מוציאה כסף מרשת הברק לרשת הביטקוין, כלומר סוגרת ערוץ – קוראים לה טרנזקציה "בין שכבות", מכיוון שהיא מבצעת אינטראקציה בין שתי הרשתות.
- אם הטרנזקציה נועדה להישאר ברשת הברק, היא נקראת טרנזקציית "שכבה שנייה". למרות שאם טרנזקציית שכבה שנייה משודרת לרשת הביטקוין (בטעות, בזדון, או כתגובה לצד השני), היא הופכת לטרנזקציית בין-שכבות. המהות של רשת הברק היא לוודא שטרנזקציות שכבה שנייה לא יהיה ניתן למחוק.

בעצם, טרנזקציות שנוצרו ברשת הברק ניתן לשדר באופן לגיטימי לרשת הביטקוין, כלומר טרנזקציות שכבה שנייה הן טרנזקציות שכבה ראשונה עם ייחוד מסויים. רשת הביטקוין היא איטית ורשת הברק צריכה להיות מהירה, ולכן התכנית של רשת הברק נועדה לגרום לכך שהשימוש בטרנזקציות שכבה שנייה בטרנזקציות שכבה ראשונה (כלומר, שידורן לרשת הביטקוין) יהיה נדיר ככל האפשר.

שכלול סקריפט הנעילה (Locking Script)

קודם לכן, כל סקריפטי הנעילה המעורבים השתמשו רק בריבוי-חתימות. כעת נראה אם אנחנו יכולים למצוא דרך בטוחה לעדכן את המאזנים ע"י שינוי ה-Outputים שבטרנזקציות שאנחנו בונים. המטרה היא פשוטה, נממש עונש נגד רמאות, שמוגדרת כמי שמנסה לשדר טרנזקציה ישנה. מבלי לצלול לדרכים השונות בהן ניתן להעניש משתתפית, מה שרשת הברק עושה זה לגרום לרמאיז לאבד את כספו, ברגע שהצד השני שם לב לניסיון ההונאה, בתוך פרק זמן מוגבל (בתוכנות ארנק רגילות זה לא יכול לקרות אלא אם כן יש באג חמור – אבל מי שרוצה לפעול במכוון באופן זדוני יכולה לכתוב תוכנה כזו).

בחזרה לדוגמא שלנו, אם אליס משדרת את TX A1, שנותנת לה 0.5 ביטקוין, אנחנו רוצים להכניס בתכנית מנגנון שיאפשר לבוב מספיק זמן כדי לשים לב לזה ולפעול כדי לקחת לאליס, כעונש, את כל הכסף.

ובאן נכנסת לתמונה "[נעילת זמן](#)" (Timelock), באופן ספציפי, OP_CHECKSEQUENCEVERIFY. נעילת זמן בעצם מתחילה ספירה לאחור מהרגע שטרנזקציה משודרת (או יותר נכון, מהרגע שהכורזת מוסיפה אותה לבלוק), ונועלת את הכספים עד אשר תקופת הזמן המצוינת חלפה. אם טרנזקציה שכוללת נעילת זמן של 40 יום שודרה היום, לא ניתן להוציא ממנה כספים עד שחלפו 40 יום. אם היא תשודר רק בעוד שנה מהיום, לא ניתן יהיה להוציא ממנה כספים עד שתחלוף שנה + 40 יום (מהיום). נעילת הזמן היחסית הזו יכולה לעזור לנו בבניית סקריפט נעילה חדש:

- אליס יכולה לשדר את TX A1 כדי לקבל את כספה חזרה אם בוב מפסיק להגיב.
- היא צריכה לחכות שבועיים לפני שתוכל לבזבז את ה-UTXO של TX A1 אם הוא משודר.

- אם בוב דווקא מגיב ומגלה שאליס שידרה את TX A1, הוא יכול לקחת את כל הכסף מייד, כעונש לאליס.

טרנזקציית ההתחייבות TX A1 המשודרגת נראית ככה:

TX A1(Commitment Transaction 02) Only Alice can broadcast

Output 0
Amount: 0.5 BTC

Locking script

- If two weeks have passed since this transaction is broadcasted, and Alice's signature is presented, then Alice gets the money.
- Else if Bob's signature is presented, then Bob gets the money.

אם אليس מנסה ליצור טרנזקציה בין-שכבות על בסיס TX A1, היא לא תוכל לקבל את הכסף עד אשר חלפו שבועיים, שנאכפים ע"י נעילת הזמן (Timelock). באותו זמן, אם בוב (או ליתר דיוק, מחשב הצומת של בוב) מגלה ש-TX A1 שודרה, הוא יכול מייד לבזבז את ה-0.5 ביטקוין שיש בה (מחשב הצומת של בוב יעביר את הסכום לארנק הפרטי של בוב).

התכנית הנ"ל מוודאת שאם טרנזקציה ישנה שודרה, הכסף של אليس יילקח ממנה כעונש. מצד שני, התכנית נותנת יותר מדי כוח לבוב, שבאמצעות החתימה שלו יכול לקחת את כל הכסף, מה שמעודד אותו להפוך להיות האיש הרע ([בשהאודהנט שלך הופכית לשונאית שלך](#)). הכוונה המקורית במתן אפשרות לאליס לקבל חזרה את ה-0.5 ביטקוין שלה, היא כדי להגן עליה במקרה שבו מפסיק להגיב. לעומת זאת, לבוב יש עכשיו מוטיבציה לרמות את אלים כדי שתשדר את הטרנזקציה, בכך שבכוונה יפסיק להגיב - ואז בשאליס תשדר את הטרנזקציה, בוב מייד ייקח את כל הכסף. עלינו לשכלל את התכנית.

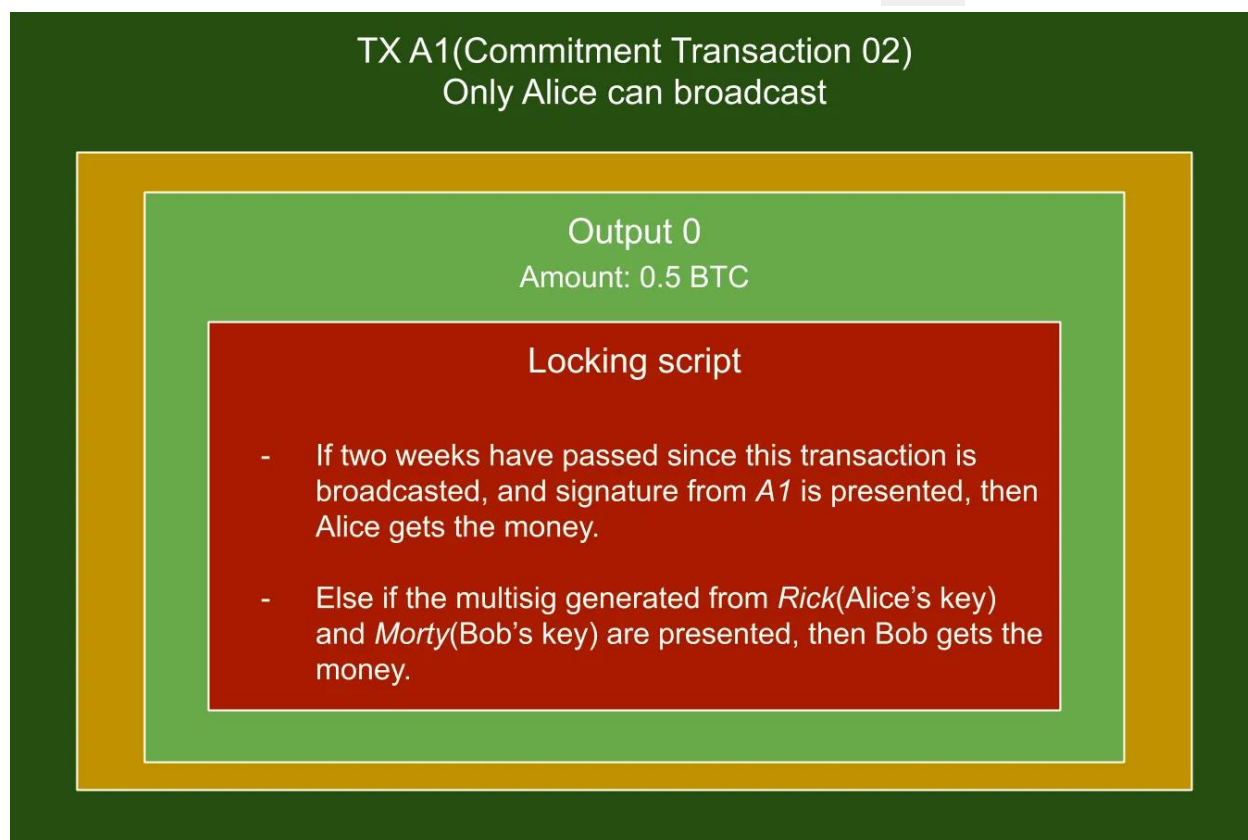
הרבה מפתחות, הרבה בלבול - ההיסטוריה של RSMC

כשאנו מדברות על "החתימה של אליס", אנחנו בעצם מתכוונות לחתימה שנוצרה ע"י אחד המפתחות שאליס שולטת בהם. הרעיון של "חתימה של אליס" הוא למעשה חתימה באמצעות מפתח פרטי. מצד שני, אליס יכולה ליצור כמה מפתחות פרטיים שהיא צריכה, ולא רק אחד. היכולת הזו חשובה כי היא מאפשרת לבטל את היתרון של בוב בשיטה הקודמת.

במקום לבקש רק את החתימה של בוב בסקריפט הנעילה, נבקש ריבוי-חתימות (multisig) כדי למנוע מבוב לגנוב את הכסף. כשהמשתתפות מסכימות לעדכן את המאזנים ע"י יצירת טרנזקציות חדשות, הן ישתפו בינן את המפתחות הפרטיים בריבוי-החתימות של הטרנזקציות הישנות, כלומר יאפשרו לצד השני לשחרר נעילות מסוימות שהוא לא יכול היה לשחרר קודם.

לצורך ההדגמה, ניתן למפתחות הפרטיים שמות כדי לעקוב אחריהם. בנקודת ההתחלה, אליס תיצור ארבעה מפתחות פרטיים: **Alice**, **Rick**, **Batman**, ו-**Tom**. בוב, בהתאמה, ייצור מפתחות פרטיים: **B1**, **Morty**, **Robin**, ו-**Jerry**.

מצד אליס, הטרנזקציה TX A1 נראית ככה:



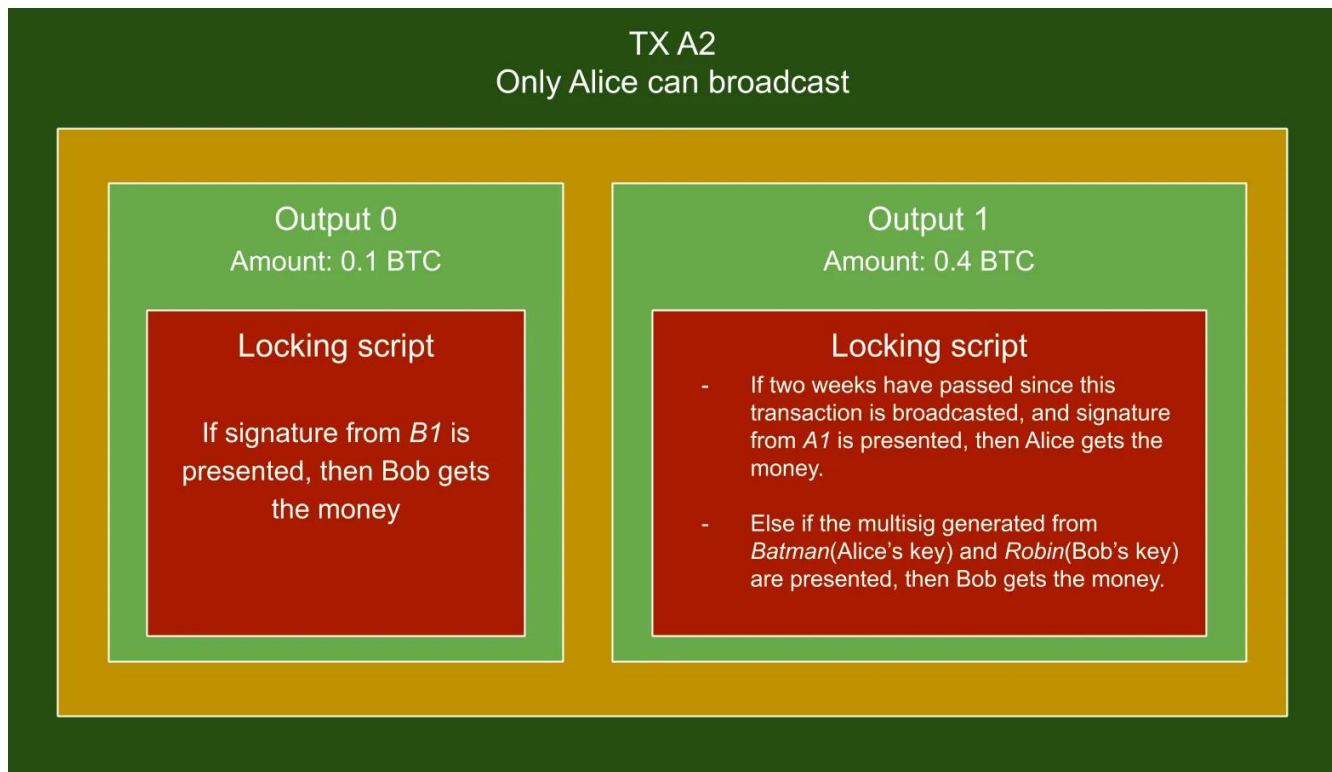
מהצד של בוב, הטרנזקציה TX B1 היא יותר פשוטה, מכיוון שאף פעם אין לו אינטרס לשדר אותה (היא נותנת את כל הכסף לאליס):



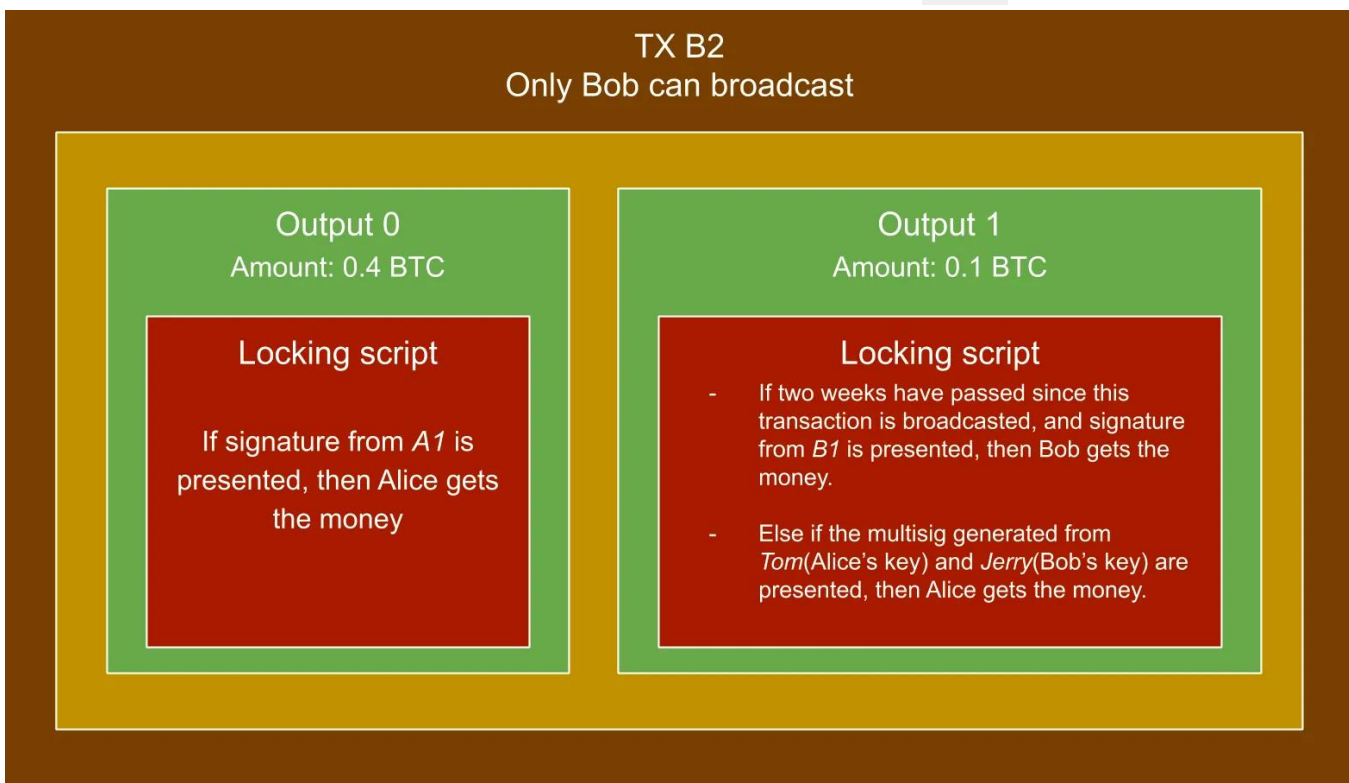
ע"י שימוש בריבוי-חתימות ב-TX A1, התבטל היתרון של בוב. אם לא היו עדכונים למאזן, כלומר TX A2 ו-TX B2 עדיין לא נוצרו, הכסף של אליס נשאר מאובטח אפילו אם בוב מפסיק להגיב (בטעות או בזדון) - מכיוון שלבוב אין את המפתח Rick.

אם הק רוצים עדכונים נוספים של המאזנים, הק צריכים ליצור טרנזקציות חדשות עם סקריפט נעילה דומה. במהלך היצירה של הטרנזקציות החדשות, הק גם יחליפו ביניהם את המפתחות הפרטיים שהשתמשו בהם בטרנזקציות הקודמות. במקרה שלנו, אליס תשלח לבוב את המפתח הפרטי Rick כאשר היא יוצרת את TX A2. צעד זה משנה את המשחק, מכיוון שהוא מבטל לחלוטין כל אינטרס של אליס לנסות לרמות ולשדר את TX A1 - אם היא תעשה זאת היא תאבד את כל כספה לבוב מכיוון שבוב ישתמש ב-Rick וב-Morty כדי לשחרר את סקריפט הנעילה.

מהצד של אליס, טרנזקציית TX A2 תראה כך:

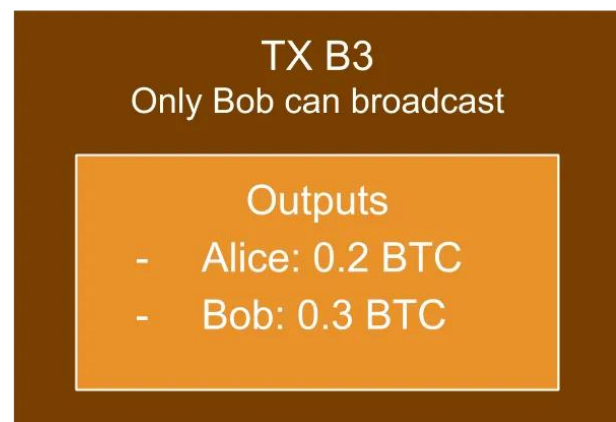
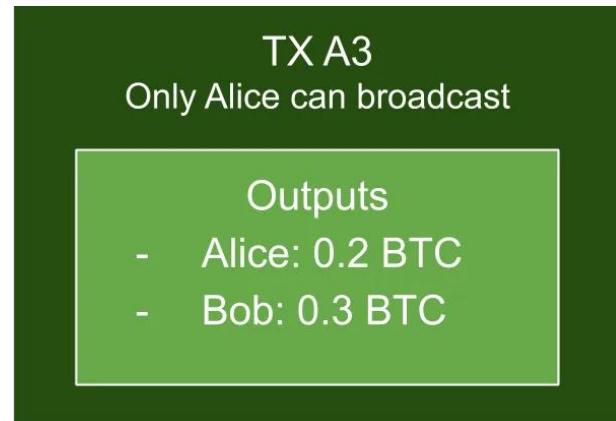


ומהצד של בוב, טרנזקציית TX B2 תראה כך:



אם אליס תחליט ליצור עדכון נוסף במאזן עבור תשלום לבוב של 0.2 ביטקוין, הצעדים יהיו כדלקמן:

1. ליצור טרנזקציות חדשות. גם אליס וגם בוב ייצרו מפתחות חדשים כדי ליצור טרנזקציות דומות לאיור האחרון. הטרנזקציות החדשות, TX A3 ו-TX B3, ייתנו לאליס 0.2 ביטקוין ולבוב 0.3 ביטקוין.
2. לאחר שיתוף הטרנזקציות החדשות, הק יחליפו ביניהם את המפתחות הפרטיים הישנים. אליס תיתן לבוב את המפתח הפרטי Batman, ובוב ייתן לאליס את המפתח הפרטי Jerry. באופן זה, אף אחת לא תרצה לשדר את הטרנזקציות הישנות, מכיוון שזה יאפשר לצד השני לקחת מייד את כל הכסף.



כמו קודם, לבוב אין אינטרס לשדר את הטרנזקציה הישנה TX B2. אם אליס מנסה לרמות ע"י שידור TX A2, היא תצטרך לחכות שבועיים לפני שהיא תוכל להשתמש בכסף, אבל בוב, שכבר קיבל ממנה את המפתח הפרטי Batman (כשהק עדכנו את המאזן בינק), וכבר היה לו מראש את המפתח הפרטי Robin שהוא עצמו יצר, יוכל לקחת מייד את הכסף בלי לחכות שבועיים. זה האינטרס של אליס לא לשדר טרנזקציה ישנה, ובוב חייב לבדוק את הטרנזקציות בבlookצ'יין כדי לבדוק אם אליס מנסה לרמות או לא. הבעיה נפתרה: כל עוד כולק פועלית מתוך האינטרס האישי שלהם, אף אחת לא תנסה לרמות. המנגנון הכלכלי מבוסס-התמריצים המתוחכם הזה נקרא RSMC: Revocable Sequence Maturity Contract, והוא ההשראה לפיתוח של רשת הברק.

כמו שהשם מרמז, החוזה (Contract) הוא ניתן-לביטול (Revocable) מכיוון שטרנזקציות שכבה שנייה ניתן להחליף בטרנזקציות חדשות יותר. הוא משתמש בנעילת-זמן על בסיס OP_CHECKSEQUENCEVERIFY, כלומר יש פה מספר-מעקב (Sequence), שמייצג את הזמן בבולקצ'יין (כמו גובה בלוק או timestamp). מכיוון שהכסף נעול בזמן, הכסף צריך להגיע ל"בשלות" (Maturity) כדי שניתן יהיה להוציא אותו כשמשפיק זמן עבר. ביחד, השיטה הזו קיבלה את השם חוזה מבטיל ניתן-לביטול עם מספר-מעקב (Revocable Sequence Maturity Contract). ל-RSMC יש את המגבלות שלו, מכיוון שהוא יכול לשרת רק שני משתתפים בכל פעם. אנחנו רוצים להרחיב את הרשת כדי שאנשים יוכלו לבצע תשלומים בצורה חלקה. עם שכלול קטן ל-RSMC, נגיע לבסיס של רשת הברק, שנקרא HTLC: Hashed Time Lock Contract - חוזה נעילת-זמן מגובב.

סיכום סדר הפעולות בשיטת RSMC (מומלץ להיעזר באיורים כשעוברת על הסעיפים):

1. אליס מייצרת מפתח פרטי A1 ושולחת לבוב את המפתח הציבורי שלו.
2. בוב מייצר מפתח פרטי B1 ושולח לאליס את המפתח הציבורי שלו.
3. אליס מייצרת טרנזקציית תקצוב עם Output מסוים שיש בו 0.5 ביטקוין, וכדי להוציא אותם צריך גם חתימה מתאימה ל-A1 וגם חתימה מתאימה ל-B1. אליס עדיין לא משדרת את הטרנזקציה, אלא שולחת לבוב רק את ה-ID של הטרנזקציה וה-Output Number הרלוונטי.
4. בוב מייצר מפתח פרטי Morty ושולח לאליס את המפתח הציבורי שלו.
5. אליס מייצרת מפתח פרטי Rick, ובונה את TX A1, שמבזבזת את טרנזקציית התקצוב (כל טרנזקציות ההתחייבות מעכשיו יבזבזו את ה-Output המסוים של טרנזקציית התקצוב). למרות שיש לאליס את A1, היא עדיין לא מצרפת את החתימה שלה. אליס שולחת את הטרנזקציה לבוב.
6. בוב בודק את הפרטים של הטרנזקציה TX A1 ושולח לאליס את החתימה שלו לטרנזקציה (באמצעות B1).
7. אליס משדרת את טרנזקציית התקצוב, כי היא יודעת שיש לה את TX A1 עם החתימה של בוב, ואם היא רוצה לסגור את הערוץ היא יכולה לצרף את החתימה שלה (באמצעות A1), לשדר את הטרנזקציה ולקבל את ה-0.5 ביטקוין בחזרה.
8. לצורך הסימטריה, בוב רוצה שתהיה לו TX B1. הוא בונה את הטרנזקציה ושולח את הפרטים שלה לאליס.
9. אליס מסתכלת על הפרטים של הטרנזקציה TX B1 ושולחת לבוב את החתימה שלה (באמצעות A1). אין לה שום בעיה לחתום עליה, כי הטרנזקציה הזו נותנת לה את כל ה-0.5 ביטקוין שבערוץ. גם בוב יכול עכשיו לסגור את הערוץ באופן חד צדדי, ע"י צירוף החתימה שלו ל-TX B1 באמצעות B1, ושידור הטרנזקציה (טכנית אין לו סיבה אף פעם לעשות את זה כי אליס תקבל את כל ה-0.5 ביטקוין).
10. שני הצדדים מחכים שטרנזקציית התקצוב תעבור מספיק confirmations. בתום שלב זה, הערוץ פתוח וניתן להתחיל להשתמש בו לביצוע תשלומים, לפי הצעדים המפורטים להלן.
11. אליס רוצה לשלם 0.1 ביטקוין לבוב דרך הערוץ.

12. אליס מייצרת שני מפתחות פרטיים: Tom ו-Batman, ושולחת לבוב את המפתחות הציבוריים שלהם.

13. בוב מייצר שני מפתחות פרטיים: Robin ו-Jerry, ושולח לאליס את המפתחות הציבוריים שלהם.

14. אליס בונה את טרנזקציית ההתחייבות TX A2 ושולחת אותה לבוב, ללא חתימות.

15. בוב בודק את הפרטים של הטרנזקציה TX A2, ושולח לאליס את החתימה שלו (באמצעות B1).

16. בוב בונה את טרנזקציית ההתחייבות TX B2 ושולח אותה לאליס, ללא חתימות.

17. אליס מסתכלת על הפרטים של הטרנזקציה TX B2, ושולחת לבוב את החתימה שלה (באמצעות A1).

18. אליס רוצה להבטיח לבוב שהיא לא תנסה לחתום ולשדר את טרנזקציית ההתחייבות הישנה TX A1. היא שולחת לבוב את המפתח הפרטי Rick. אם אליס תנסה לרמות, בוב ישתמש ב-Rick שהיא נתנה לו וב-Morty שהוא עצמו יצר, כדי לקחת לעצמו מייד את כל הכסף בערוץ. אם בוב רוצה לסגור את הערוץ באופן חד צדדי, הוא יעשה את זה ע"י שידור TX B2, החתימה שאליס נתנה לו בסעיף 17, וחתימה שהוא יכול לייצר (באמצעות B1). לכן מבחינת בוב התשלום הסתיים.

19. לאליס לא אכפת שבוב ישדר את TX B1 שנותנת לה את המקסימום בערוץ (0.5 ביטקוין) – יותר ממה שמגיע לה (0.4 ביטקוין אחרי התשלום). אם אליס רוצה לסגור את הערוץ באופן חד צדדי, היא תעשה את זה ע"י שידור TX A2, החתימה שבוב נתן לה בסעיף 15, וחתימה שהיא יכולה לייצר (באמצעות A1). לכן גם מבחינת אליס התשלום הסתיים.

20. אם הק רוצה לבצע תשלום נוסף:

a. כל אחת ייצור שני מפתחות פרטיים חדשים וישלח לשניה את המפתחות הציבוריים שלהם.

b. אליס תבנה את TX A3 ותשלח אותה לבוב ללא חתימות. בוב יבדוק את הטרנזקציה ויענה לה עם החתימה שלו (באמצעות B1).

c. בוב יבנה את TX B3 וישלח אותה לאליס ללא חתימות. אליס תבדוק את הטרנזקציה ותענה לו עם החתימה שלה (באמצעות A1).

d. אליס תשלח לבוב את המפתח הפרטי Batman. אם היא תנסה לרמות ולשדר את TX A2 עם חתימות, בוב יגיב עם Batman (שהיא נתנה לו) ועם Robin (שהוא עצמו יצר) וייקח את כל הכסף לעצמו מייד.

e. בוב ישלח לאליס את המפתח הפרטי Jerry. אם בוב ינסה לרמות ולשדר את TX B2 עם חתימות, אליס תגיב עם Tom (שהיא עצמה יצרה) ועם Jerry (שבוב נתן לה) ותיקח את כל הכסף לעצמה מייד.

חזרה נעילת-זמן מגובב

דמיינו שיש לנו שלושה משתתפים ושני ערוצים. ערוץ אחד פתוח בין אליס ובוב, והערוץ השני פתוח בין בוב לצ'רלי. אם אליס רוצה לשלוח כסף לצ'רלי, יש לה שתי אפשרויות:

1. אליס וצ'רלי יכולים לפתוח ערוץ בינן ולבצע את הטרנזקציה

2. אליס וצ'רלי יכולים להשתמש בערוצים הקיימים

נראה שהגישה הראשונה תעבוד, אבל היא לא תפתור את הבעיה כשמספר המשתתפים גדל. החשבון הוא פשוט:

- עבור 3 משתתפים אנחנו צריכים 3 ערוצים
- עבור 4 משתתפים אנחנו צריכים 6 ערוצים
- עבור 10 משתתפים אנחנו צריכים 45 ערוצים
- עבור 1000 משתתפים אנחנו צריכים 499,500 ערוצים!

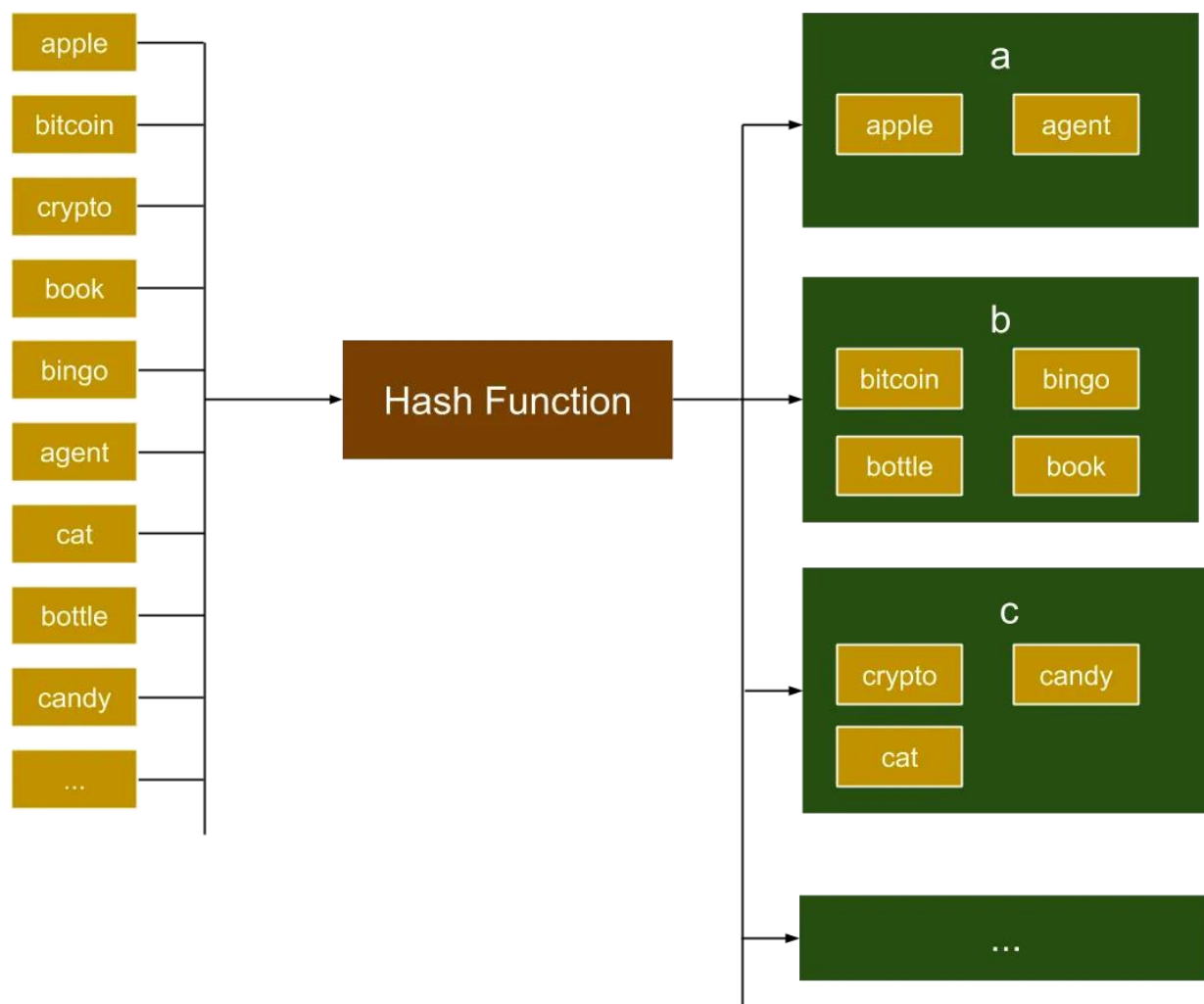
זה פשוט יותר מדי ערוצים! בהשוואה לאלטרנטיבה שבה אליס וצ'רלי מבקשים מבוב לעזור, הגישה הראשונה היא פחות מועדפת ופחות ניתנת ליישום. הק כבר בנו ערוץ עם בוב, למה לא "לנתב" כסף דרך בוב הלוח וחזור?

פונקציית גיבוב (Hash) קריפטוגרפית

הערת מתרגם: מי שמכירה יכולה לדלג על פרק זה.

כדי לממש ניתוב בין ערוצים, אנחנו צריכים לקחת בהשאלה כלי קריפטוגרפי שנקרא פונקציית גיבוב, ולהבין את המאפיינים שלה.

פונקציית גיבוב ממפה כל קלט לפלט בגודל קבוע. לדוגמא, פונקציית גיבוב פשוטה יכולה להיות לקחת מילים ולקבץ אותן לפי האות הראשונה.



הקלט הוא מילים (apple, agent), והפלטים הם אותיות (a). פונקציית גיבוב נוספת יכולה להיות פעולת ה-Modulo (מציאת שארית מחלוקה במספר). לדוגמא, אנחנו יכולים להגדיר פונקציית גיבוב שיכולה לקבל כ-input כל מספר שלם, ולחשב את השארית מחלוקה ב-7. לכל הפלטים יש גודל קבוע: מספר בעל ספרה אחת בין 0 ל-6.

Input(divident)	Hash Function	Output(remainder)
0	$0 \% 7$	0
1	$1 \% 7$	1
2	$2 \% 7$	2
3	$3 \% 7$	3
4	$4 \% 7$	4
5	$5 \% 7$	5
6	$6 \% 7$	6
7	$7 \% 7$	0
8	$8 \% 7$	1
...	$\dots \% 7$	0 - 6

פונקציית גיבוב קריפטוגרפית נדרשת לקיים את שתי התכונות הבאות:

- **פונקציה חד-כיוונית** (One-Way Function) – כלומר בהינתן פלט, קשה למצוא את הקלט. אם הפלט הוא 1, קשה לדעת אם הקלט הוא 1, 8, או כל מספר שבחלוקה ב-7 נותן שארית 1.
 - **מניעת התנגשות** (Collision Resistance) – כלומר זה קשה למצוא שני קלטים שנותנים את אותו פלט. לפונקציית הגיבוב הנ"ל דווקא אין את התכונה הזו, וקל למצוא למשל את הקלטים 2 ו-9 שנותנים את אותו פלט 2, או את שתי המילים apple, agent שהפלט עבורן הוא האות a. פונקציות גיבוב קריפטוגרפיות משיגות את התכונות הללו באמצעות שימוש בפונקציות מתמטיות מורכבות. מבלי לצלול לתוך "מחילת הארנב", האינסטינקט מאחורי התכנון שלהן הוא קל להבנה. כדי לבנות פונקציה חד-כיוונית, נדרשת פונקציה מתמטית שקל לוודא אותה כשיודעית את התשובה, אבל קשה לפתור אותה. או ליתר דיוק, יותר קשה למצוא תשובה מאשר לבדוק תשובה. בקריפטוגרפיה, "להיות קשה" משמעותו לדרוש יותר כוח חישוב וזמן. פירוק לגורמים זו דוגמה טובה:
 - אתגר ראשון: למצוא שני מספרים שהמכפלה שלהם היא 2449.
 - אתגר שני: לבדוק ש-31 כפול 79 נותן 2449.
- לא קשה להסיק שיותר קל לפתור את האתגר השני מאשר את הראשון. למרות שקריפטוגרפיה מתבססת על מודלים מתמטיים מורכבים (לדוגמא [discrete logarithm](#)), העיקרון הזה נשאר ללא שינוי. מצד שני, כדי להשיג מניעת התנגשות, הגודל של פלטים ייחודיים חייב לגדול משמעותית, כדי להקטין את הסיכוי שקלטים שונים יתנגשו ויתנו את אותה תוצאה. אחד האלגוריתמים הנפוצים ביותר, SHA256, נותן 2^{256} פלטים אפשריים, שזה מספר **אסטרונומי מעבר לכל דמיון**.

דוגמה לשימוש בפונקציות גיבוב קריפטוגרפיות היא להסתיר סוד ואח"כ לחשוף אותו ([הוכחה באפס ידיעה](#)). דוגמא ביזארית יכולה להיות שאלים טוענת שהיא האדם הראשונה בעולם שיודעת ידיעה קסומה שיכולה להפוך אנשים למיליארדרות, אבל צ'רלי לא מאמין לה. אם אלים תספר לצ'רלי את הידיעה ישירות, בוב יקפוץ מייד ויטען שהוא ידע את הידיעה הזו קודם. כדי למנוע את גניבת זכויות היוצרות, אלים תפעיל פונקציית גיבוב קריפטוגרפית על הידיעה, תשתף את התוצאה עם בוב, ותטען שהיא יודעת את הידיעה הסודית מבלי לחשוף אותה.

היא תשתמש ב-SHA256 כדי [לגבב את הפטנט הסודי](#), [שמייצר את הפלט](#):

8816cee3feb85ccaado557ffb2f6b38947a27bea4ccdea1835fdded53ad71c31a

אליס אז תשתף את התוצאה עם צ'רלי. צ'רלי יכול עכשיו לבקש מבוב להראות לו את ההודעה המקורית שמתאימה לפלט הזה. כמובן שבוב לא יודע את התשובה, ולא יכול לזייף אותה. כעת זה בטוח לאליס לחשוף את הידיעה הסודית:

money is a social construct

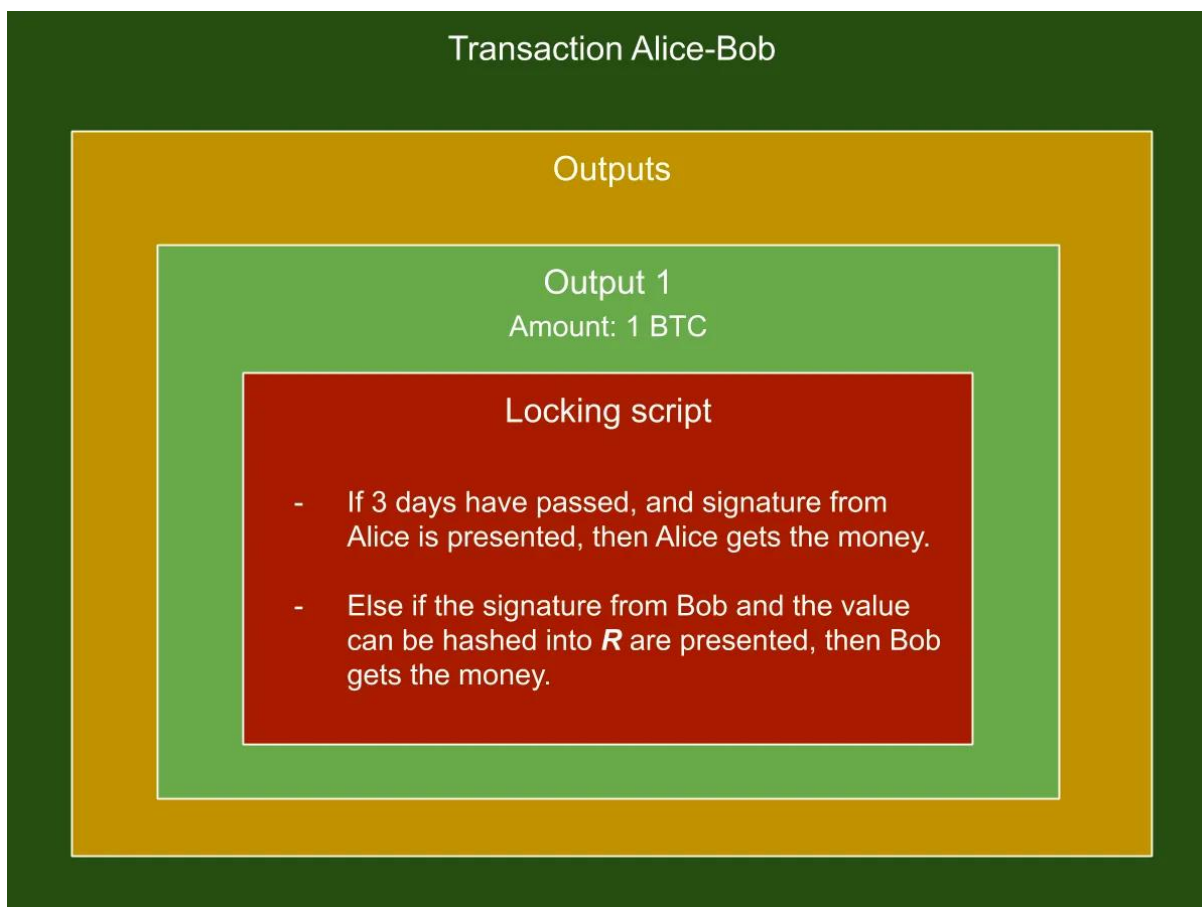
תוכנית נאיבית – הסתרה וחשיפה של סוד

זו תהיה נאיביות פשוט לבקש מבוב להעביר את הכסף אם אלים רוצה לשלוח 1 ביטקוין לצ'רלי. לבוב יהיו את כל הסיבות לקחת את הכסף לעצמו בלי להעביר כלום לצ'רלי. כדי לפתור את הבעיה, פונקציית גיבוב קריפטוגרפית נכנסת לתכנית. אם אלים רוצה לשלוח 1 ביטקוין לצ'רלי, זהו צ'רלי שצריך להתחיל ב-"בקשת תשלום". הינה מה שנעשה:

1. צ'רלי ייצר מספר רנדומלי r וישמור אותו לעצמו כסוד (נקרא גם Preimage).
2. צ'רלי יפעיל פונקציית גיבוב על הסוד r , וישלח את התוצאה R לבוב יחד עם בקשת התשלום "דרוש מאליס תשלום של 1 ביטקוין".
3. צ'רלי יחכה שבוב ישלם לו ביטקוין אחד. צ'רלי לא יסכים לספר לאף אחד את r עד שיקבל תשלום.

ברגע שבוב מקבל את בקשת התשלום ביחד עם הערך המגובב R , הוא יעביר אותם לאליס ויבקש ממנה שתשלם לו 1 ביטקוין, ע"י יצירת טרנזקציה עם סקריפט הנעילה הבא:

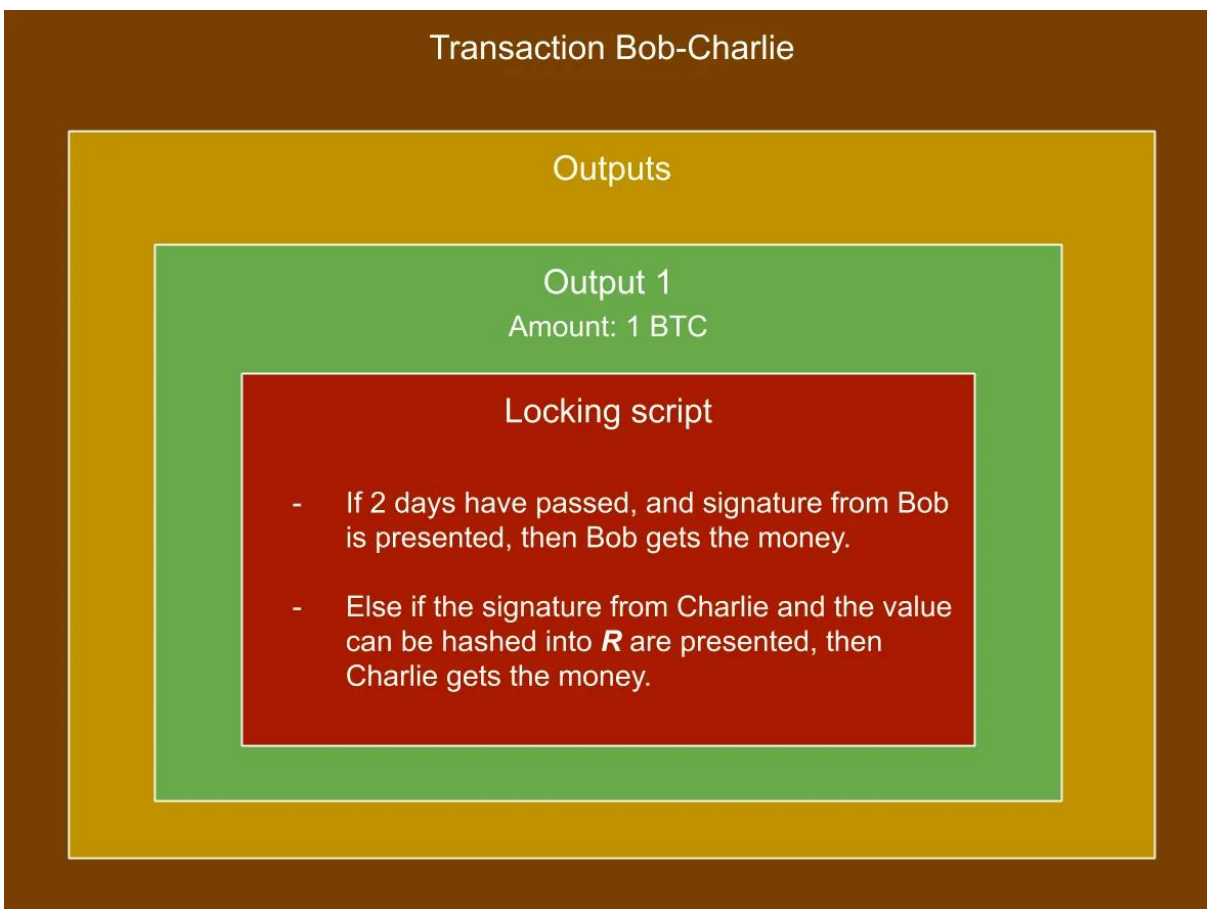
- אם עברו שלושה ימים, ונמצאת חתימה מאליס, אז אלים מקבלת את הכסף חזרה.
- אחרת, אם נמצאת גם חתימה של בוב וגם הערך שהגיבוב שלו נותן R (שהוא הסוד r), אז בוב מקבל את הכסף.



הערת מתרגם: Else- כאן הוא לא במובן של שפת-תכנות, כלומר הוא לא מבצע בדיקה ש-"חלפו שלושה ימים". למעשה אין אופרטור בסקריפט של ביטקוין שבדק דבר כזה (הסבר בהמשך). המשמעות של ה-Else כאן היא "עוד דרך לבזבז את ה-Output, בין אם חלפו או לא חלפו שלושה ימים".

אם בוב יכול לקבל את z מצ'רלי אז הוא יוכל לשחרר את הנעילה ולבזבז את ה-UTXO. אחרת, כעבור שלושה ימים, אליס תוכל לקבל חזרה את הביטקוין שלה. דרישת שלושת הימים היא דרישה בזמן אבסולוטי, בניגוד לדרישת נעילת הזמן הקודמת שעסקנו בה. היא מאפשרת לאליס לקבל את הכסף חזרה אם בוב לא יכול לספק את הסוד בתוך שלושה ימים מרגע יצירת הטרנזקציה (בלי קשר מתי היא תשודר). כעת, כשבוב בטוח שהוא יקבל 1 ביטקוין מאליס אם הוא ידע בעתיד את הערך של z , הוא יכול לשלוח לצ'רלי 1 ביטקוין ע"י יצירת טרנזקציה דומה עם סקריפט הנעילה הבא:

- אם חלפו יומיים, ונמצאת חתימה של בוב, אז בוב יכול לקבל את הכסף חזרה.
- אחרת, אם נמצאת גם חתימה של צ'רלי וגם הערך שהגיבוב שלו נותן R , אז צ'רלי מקבל את הכסף.



הערת מתרגם: ההערה על ה-Else תקפה גם כאן כמו בשרטוט הקודם.

כדי לקבל את הכסף, צ'רלי חייב לחשוף את הסוד r לבוב אחרי שידור הטרנזקציה הזו, בטרנזקציה נוספת שמבזבזת את ה-Output. אם הטרנזקציה תשודר אבל צ'רלי לא יבזבז את ה-Output, כעבור יומיים (זמן אבסולוטי) זו הופכת לטרנזקציה מסוכנת מאוד, כי שני הצדדים יכולים להתחרות על הכסף. מאידך אם צ'רלי לא ישדר את הטרנזקציה בכלל, אחרי יומיים הערוץ לא יהיה שימושי יותר – בוב לא יסכים יותר להגדלת הצד שלו במאזן בערוץ זה (לא יסכים להחלפת טרנזקציות חדשות), כי הוא יודע שצ'רלי תמיד יכול לחכות שלא יהיה פעיל ואז לנסות את מזלו לשדר את הטרנזקציה הישנה ומייד לבזבז את ה-Output. הטרנזקציה הזו טובה רק אם צ'רלי משתמש בה תוך יומיים וגם חושף את הסוד כדי לבזבז את ה-Output – אחרי יומיים היא יוצרת אי ודאות לשני הצדדים. ברגע שבו יודע את r (בהסתכלות על הבלוקצ'יין), הוא יכול להוציא את הכסף מהטרנזקציה בינו לבין אליס. כדי שלכל משתתף יהיה מספיק זמן לפעול, משך הזמן הנעול הולך וקטן ככל שמתקרבת יותר ליעד הסופי של התשלום, כלומר צ'רלי מקבל יומיים לחשוף את r , אבל בוב מקבל שלושה ימים. הפעולה של חשיפת הסוד r לאליס מבוב מוכיחה שהוא כבר שלח את הביטקוין לצ'רלי, מכיוון שזו הדרך היחידה שבה בוב יכול לגלות את הערך הסודי. אבל האם משהו יכולה לרמות?

הכסף של אליס מוגבל לחלוטין בתקופת שלושת הימים. ברגע שעברו שלושה ימים, אליס חופשית לבזבז את הכסף לא משנה אם לבוב יש את הסוד או לא. לגבי בוב, הוא יכול להוציא את הכסף רק אם הוא יודע את z. אפילו אם עברו שלושה ימים, בוב לא יכול להוציא את הכסף מבלי לדעת את z. בלי קשר לעדכון המאזנים, נראה שהתכנית הנוכחית היא סבירה אם:

- בוב מתכוון להשתמש ב-UTXO שלו ברגע שהוא יודע את z, כלומר הוא ישדר את הטרנזקציה שלו לפני שמסתיימים שלושת הימים.
 - אליס מתכוונת להשתמש ב-UTXO שלה ברגע שעברו שלושה ימים, כלומר היא תשדר את הטרנזקציה שלה מייד כשחולפים שלושת הימים (אלא אם בוב השתמש ב-UTXO לפנייה).
- אם הן לא משדרות את הטרנזקציה בזמן הקצוב, הצד השני יכול לרמות ולקחת את הכסף. אבל אם טרנזקציית שכבה־שנייה משודרת, היא הופכת לטרנזקציה בין־שכבות, מה שסותר את הדרישה שלנו מרשת הברק – להשאיר את הטרנזקציות בתוך רשת הברק ככל שניתן. לפיכך, נדרשת תוכנית חדשה עם תכונות דומות ל-RSMC – אנחנו לא רוצים להתבסס על ההוגנות של המשתתפים, אלא להעניש את מי שמתנהג באופן לא הוגן.
- עם הרקע הזה, הגיע הזמן ללמוד על התוכנית בשלמותה בחלק ב' – המימוש בפועל של רשת הברק.

הערות ומקורות ידע לפרק א'

1. [The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, Joseph Poon, Thaddeus Dryja](#)
2. [Reaching the ground with lightning, Rusty Russell](#)
3. [Cryptographic Hash Function](#)
4. [A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels, Christian Decker, Roger Wattenhofer](#)
5. [The Lightning Network Specifications](#) has all the implementation details. The actual locking scripts used are specified in [BOLT #3: Bitcoin Transaction and Script Formats](#)
6. I've set up a testing environment using [bitcoind](#) and [lnd](#) to help me understand the whole process. [Here's the gist of the trimmed log generated from lnd, which recorded how Alice and Bob opened a channel.](#)
7. [A curated list of awesome Lightning Network projects for developers and crypto enthusiasts.](#)

פרק ב' – המימוש בפועל של רשת הברק

הטרנזקציות ברשת הברק הן לא יותר משרשרת של טרנזקציות התחייבות שמצביעות על ה-**Outputs** של טרנזקציית התקצוב. רשת הברק של היום לקחה את הרעיון של RSMC והוסיפה מספר שיפורים. תחילה הופעלה [תוכנית התחייבות \(commitment scheme\)](#), בשיטת "הסתרה וחשיפה של סוד" שראינו קודם, כדי לאפשר העברה-הלאה (forward) של תשלומים. שנית, הוחלפה פעולת ריבוי-החתימות (multisig) [בהכפלת-נקודות-על-עקומה-אליפטית \(elliptic curve point multiplication\)](#).

מבנה הטרנזקציות

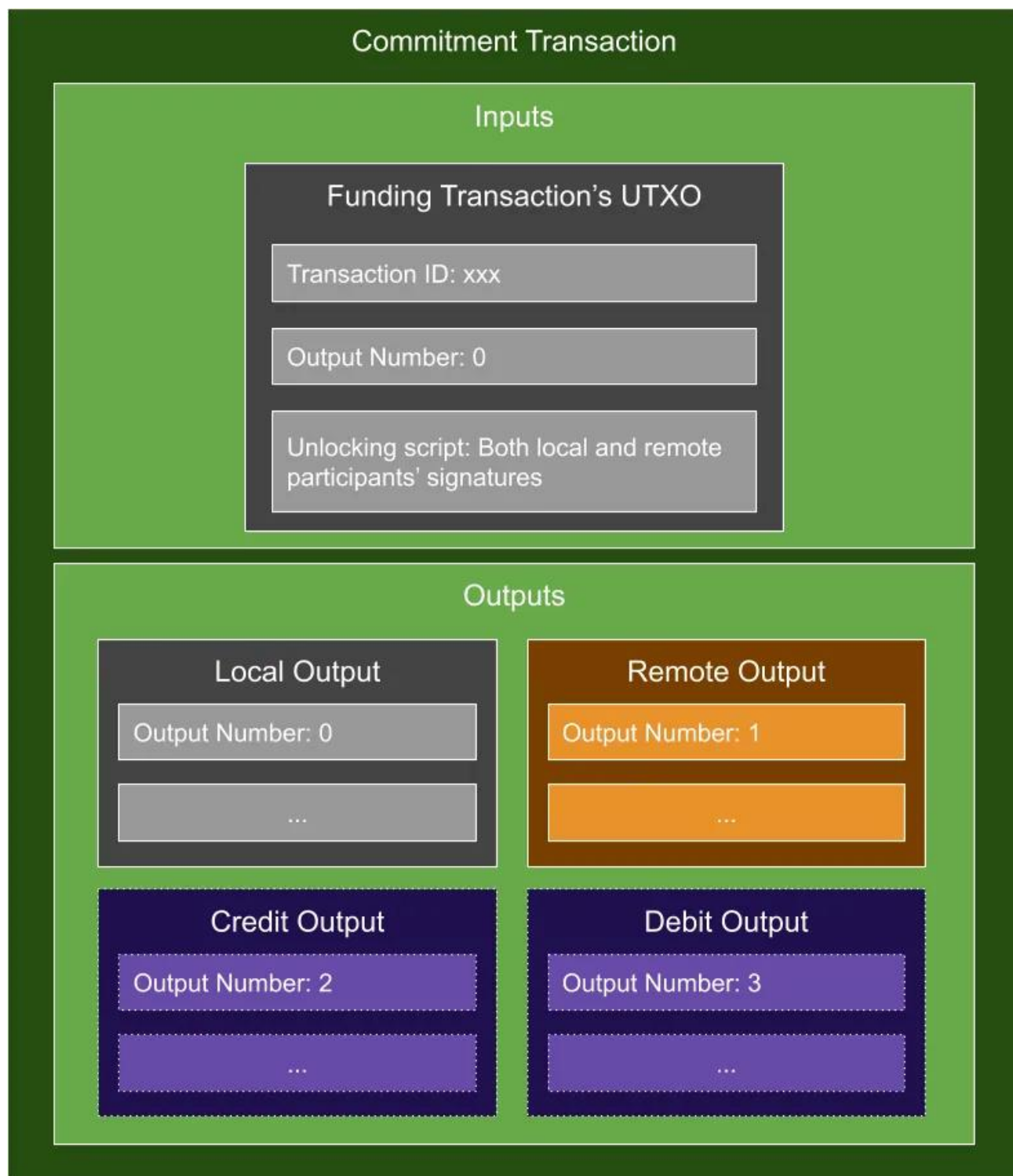
יש ארבעה סוגים של טרנזקציות ברשת הברק. שניים מהם נועדו להעביר כסף בין רשת הברק ורשת הביטקוין (טרנזקציות בין-שכבות), והשאר נועדו לעדכן את המאזנים ברשת הברק (טרנזקציות שכבה שנייה).

1. **טרנזקציית תקצוב (Funding transaction)**. טרנזקציית תקצוב נוצרת כשהערוץ נפתח. היא מעבירה כסף מרשת הביטקוין לרשת הברק ושמה את הכספים בכתובת מרובת-חתימות.
2. **טרנזקציית התחייבות (Commitment transaction)**. טרנזקציית ההתחייבות היא הבסיס של רשת הברק. בכל פעם שהמאזן משתנה, נוצרות טרנזקציות התחייבות חדשות.
3. **טרנזקציות HTLC**. טרנזקציות HTLC-Success ו-HTLC-Timeout [הונכסו במקור לתוך טרנזקציית ההתחייבות, אבל לבסוף הופרדו מהן בגרסה הסופית](#). כמו שנסביר בהמשך, אפשר להסתכל על טרנזקציות HTLC כחלק מטרנזקציית ההתחייבות, מכיוון שכל המטרה שלהן היא להוציא כספים מ-Output ימים מסויימים בטרנזקציית ההתחייבות.
4. **טרנזקציית סגירה**. טרנזקציית סגירה נוצרת כששני המשתתפים מחליטים לסגור את הערוץ בשיתוף פעולה. היא מעבירה כספים מרשת הברק לרשת הביטקוין.

טרנזקציות התחייבות כאמצעי ל-"ראיית חשבון"

לטרנזקציית התחייבות יש ארבעה סוגים של Outputs, תלוי אם היא מעבירה (forwarding) תשלומים או לא:

- Output מקומי (local), ששומר כמה כסף יש למשתתף המקומי (שמחזיק בטרנזקציה).
 - Output מרוחק (remote), ששומר כמה כסף יש למשתתף בצד השני.
 - Output זכות (credit), או "ה-HTLC הנכנס", או "ה-HTLC שהתקבל", ששומר כמה כסף המשתתף המקומי מקבל.
 - Output חובה (debit), או "ה-HTLC היוצא", או "ה-HTLC המוצע", ששומר כמה כסף המשתתף הנוכחי שולח.
- Output זכות ו-Output חובה נוצרים כאשר המשתתף יוצר או מעביר תשלום לאחרים. לדוגמא:



רשת הברק משתמשת באופן אינטנסיבי ב-"סקריפטים" של ביטקוין, כדי לאבטח את ה-Outputים ולוודא שהם ינוצלו רק באופן שהם נועדו אליו.

ביטול טרנזקציות - מפתחות ועוד מפתחות

כדי ששכבה שנייה תעבוד, היא צריכה להיות "ניתנת להחזר" (refundable) ו-"ניתנת לביטול" (revocable), כדי להתמודד עם שני המקרים הבאים:

- כשמשתתפת אחת מפסיקה להגיב לתקופה מסוימת, המשתמש השניה צריכה לקבל את כספו חזרה.

- כשטרנזקציה חדשה נוצרת, הטרנזקציה הישנה צריכה להתבטל.

טרנזקציית הביטול, שלפעמים נקראת Breach Remedy Transaction, קיימת כדי להרתיע נגד פעולות זדוניות, והיא הבסיס שמאפשר לרשת הברק להיות בטוחה.

המימוש של רשת הברק משתמש ב-"זוגות מפתחות ביטול" (revocation key pairs) במקום בריבוי-חתימות (multisig). פונקציית הגיבוב הקריפטוגרפית המעורבת היא [Elliptic Curve Digital](#)

[Signature Algorithm - ECDSA](#) ביחד עם [הכפלת נקודות עקומה אליפטית](#). התהליך הוא כדלקמן:

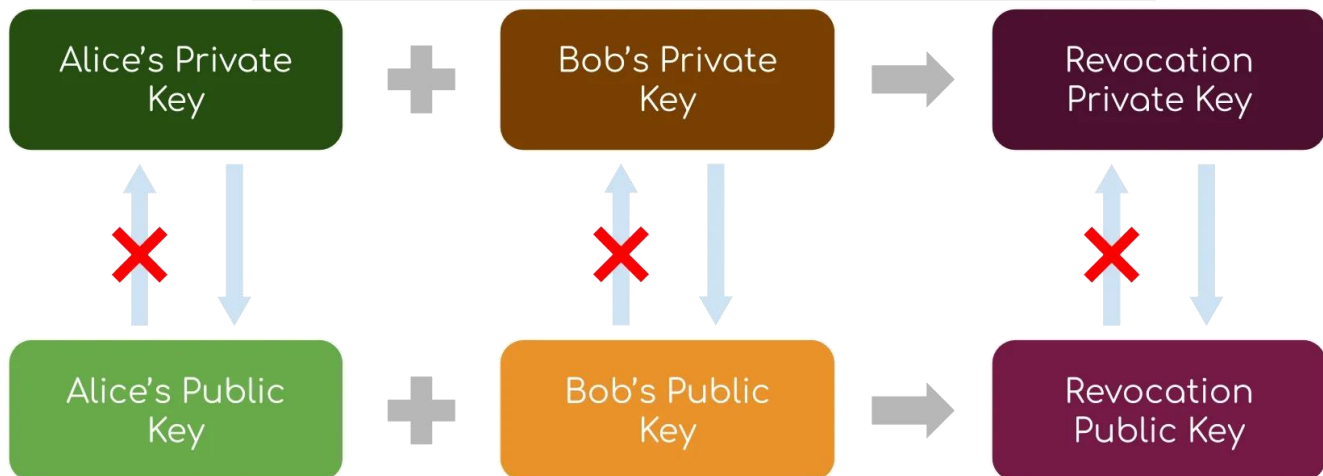
אליס ובוב, כל אחד בנפרד, יוצרים זוג מפתחות פרטי-ציבורי על בסיס ECDSA, שנקרא להם `alice_public_key`, `alice_secret_key`, `bob_public_key`, `bob_secret_key`.

מפתח ציבורי שנקרא לו `revocation_public_key` נוצר באמצעות `alice_public_key` ו-`bob_public_key`. בשביל הפשטות, אפשר להסתכל על זה כך:

$$\text{revocation_public_key} = \text{alice_public_key} + \text{bob_public_key}$$

והמפתח הפרטי המתאים לו יהיה:

$$\text{revocation_private_key} = \text{alice_private_key} + \text{bob_private_key}$$



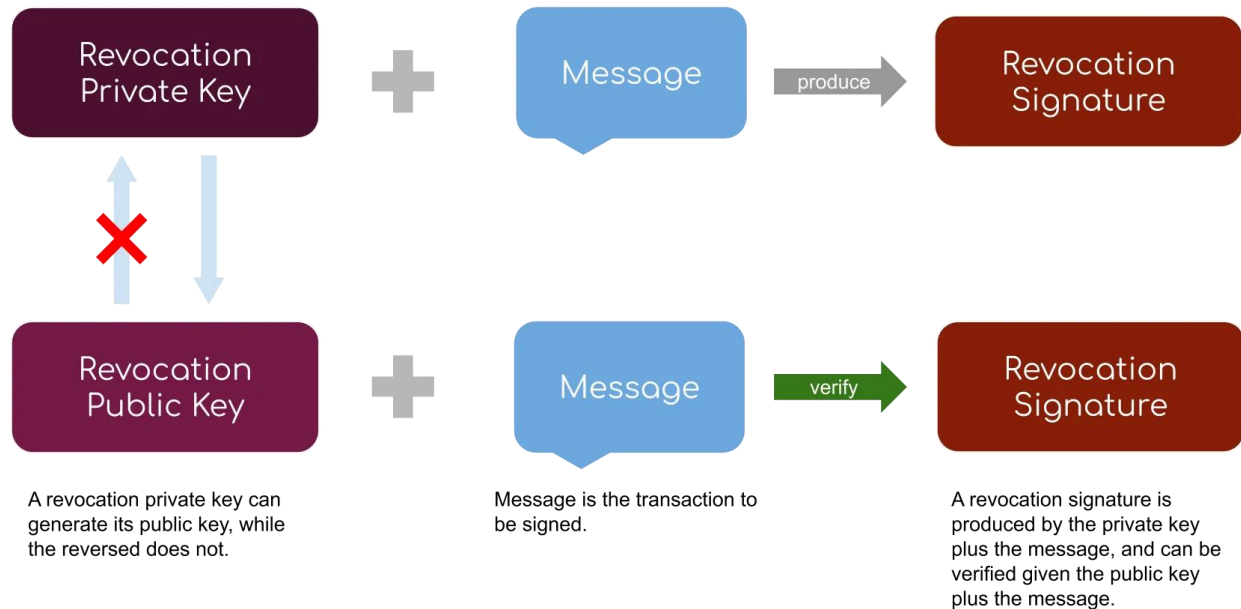
Alice's private key can generate its public key, while the reversed does not.

Bob's private key can generate its public key, while the reversed does not.

A revocation private key can generate its public key, while the reversed does not.

מכיוון שמדובר בזוג מפתחות פרטי-ציבורי, הודעות שייחתמו ע"י `revocation_private_key` ייצרו `revocation_signature`, וכל אחת יכולה לוודא את החתימה באמצעות `revocation_public_key` וההודעה.

באופן דומה לכתובת מרובת-חתימות (multisig), נדרשים שני המפתחות הפרטיים של אליס ובוב כדי לייצר חתימה תקינה. השיטה הזו עדיפה מכיוון שהיא נחשבת [יותר אינטואיטיבית ויעילה](#).



שימוש במפתחות ניתנים לביטול. ניתן להשתמש במפתח פרטי כדי לחתום מספר הודעות, ולייצר חתימות עם יחס אחד-לאחד עם ההודעות.

עם הרקע הזה, ננתח כעת לפרטים איך עובדת הפתיחה, העדכון והסגירה של ערוצים.

פתיחת ערוץ – טרנזקציית תקצוב וטרנזקציית ההתחייבות הראשונה.

התהליך לפתיחת ערוץ בין אליס לבוב של 1 ביטקוין הוא כדלקמן:

תחילה, לפני כל טרנזקציה, יש משא ומתן על ההגדרות הראשוניות של הערוץ, כמו מספר ה-`confirmations` המינימלי הדרוש עבור טרנזקציית התקצוב, והמפתחות הציבוריים של "מפתחות הביטול" (revocation keys), אשר הינם:

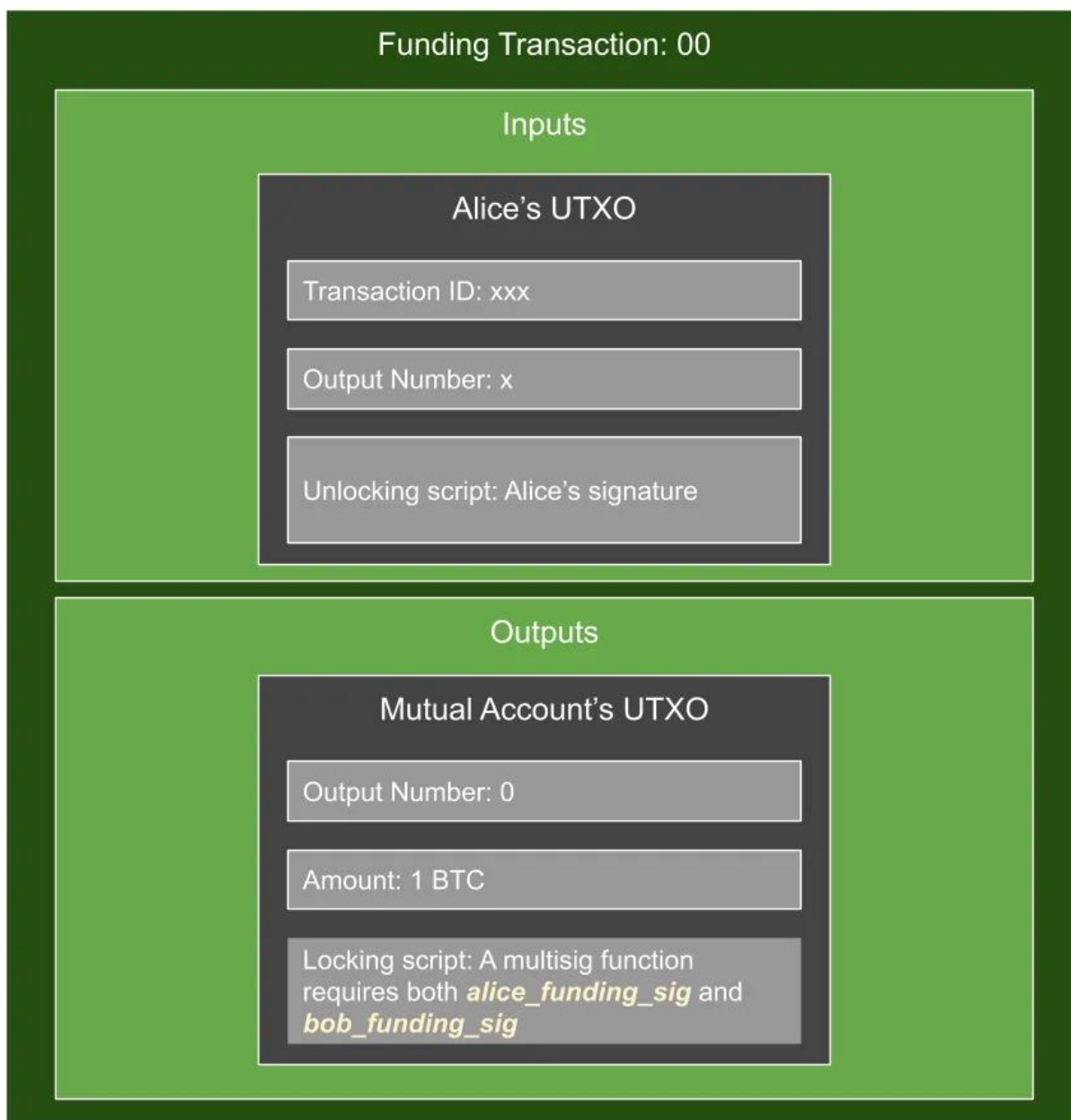
מהצד של אליס, היא יוצרת את המפתחות הציבוריים `alice_pubkey_tx_a1` ו-`alice_pubkey_tx_b1` ושולחת אותם לבוב. את המפתחות הפרטיים התואמים `alice_prikey_tx_a1` ו-`alice_prikey_tx_b1` היא שומרת לעצמה.

- מהצד של בוב, הוא יוצר את המפתחות הציבוריים `bob_pubkey_tx_a1` ו-`bob_pubkey_tx_b1` ושולח אותם לאליס. את המפתחות הפרטיים התואמים `bob_prikey_tx_a1` ו-`bob_prikey_tx_b1` הוא שומר לעצמו.
- אליס משתמשת ב-`alice_pubkey_tx_a1` ו-`bob_pubkey_tx_a1` כדי לייצר את `revocation_pubkey_tx_a1` שמאפשר לבדוק את החתימה `revocation_sig_tx_a1`. שימו

לב שלא אליס ולא בוב יכולות ליצור את החתימה הזו בינתיים. רק אם אליס תעביר לבוב את `alice_prikey_tx_a1`, אז בוב יוכל לייצר את החתימה (ורק בוב יוכל לעשות זאת).

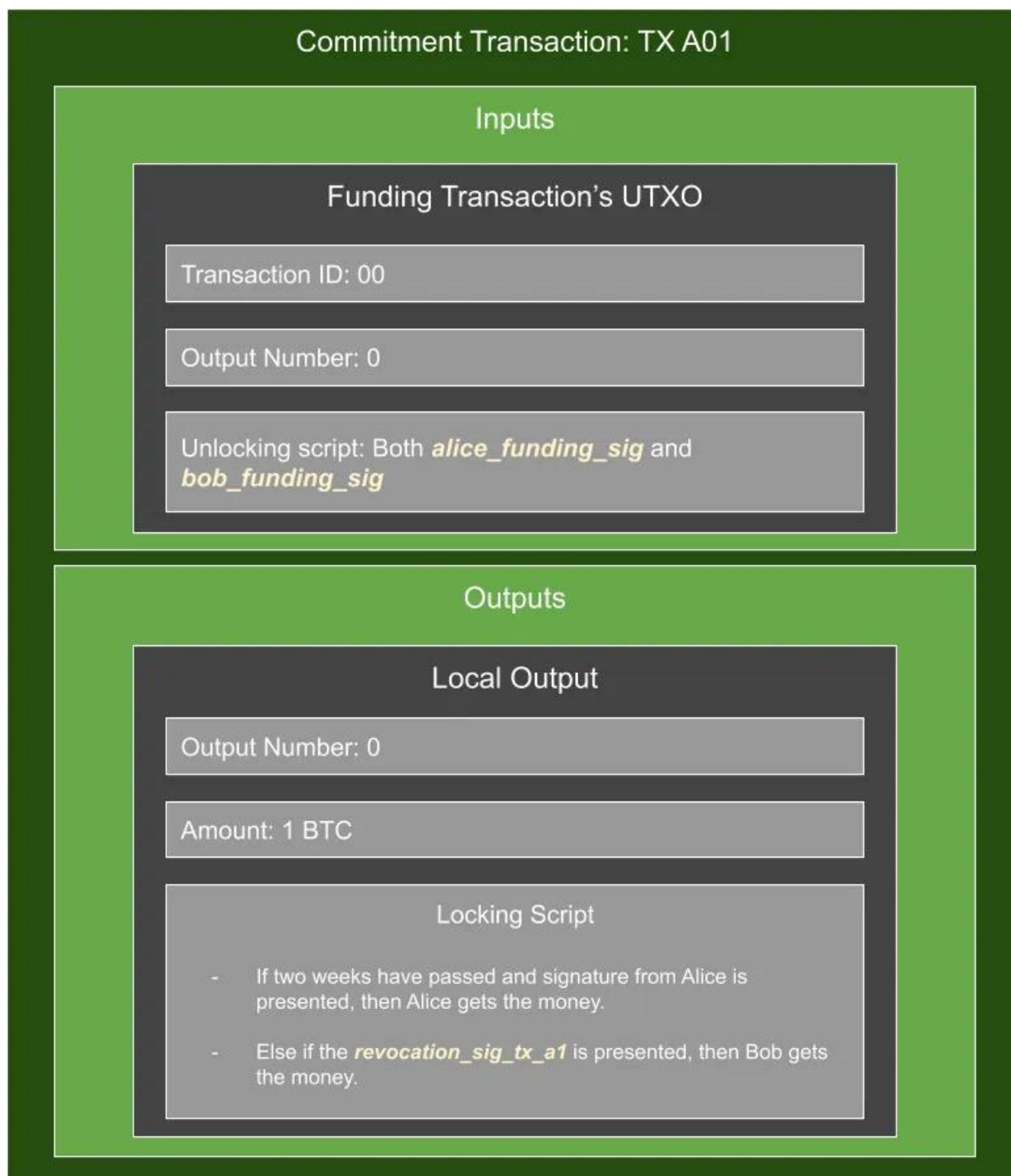
- בוב משתמש ב-`alice_pubkey_tx_b1` ו-`bob_pubkey_tx_b1` כדי לייצר את `revocation_pubkey_tx_b1` שמאפשר לבדוק את החתימה `revocation_sig_tx_b1`. שימו לב שלא אליס ולא בוב יכולות ליצור את החתימה הזו בינתיים. רק אם בוב יעביר לאליס את `bob_prikey_tx_b1` אז אליס תוכל לייצר את החתימה (ורק אליס תוכל לעשות זאת).

שנית, אליס מייצרת את טרנזקציית התקצוב, חותמת עליה, ומשתפת עם בוב את ה-ID וה-Output Number שלה.

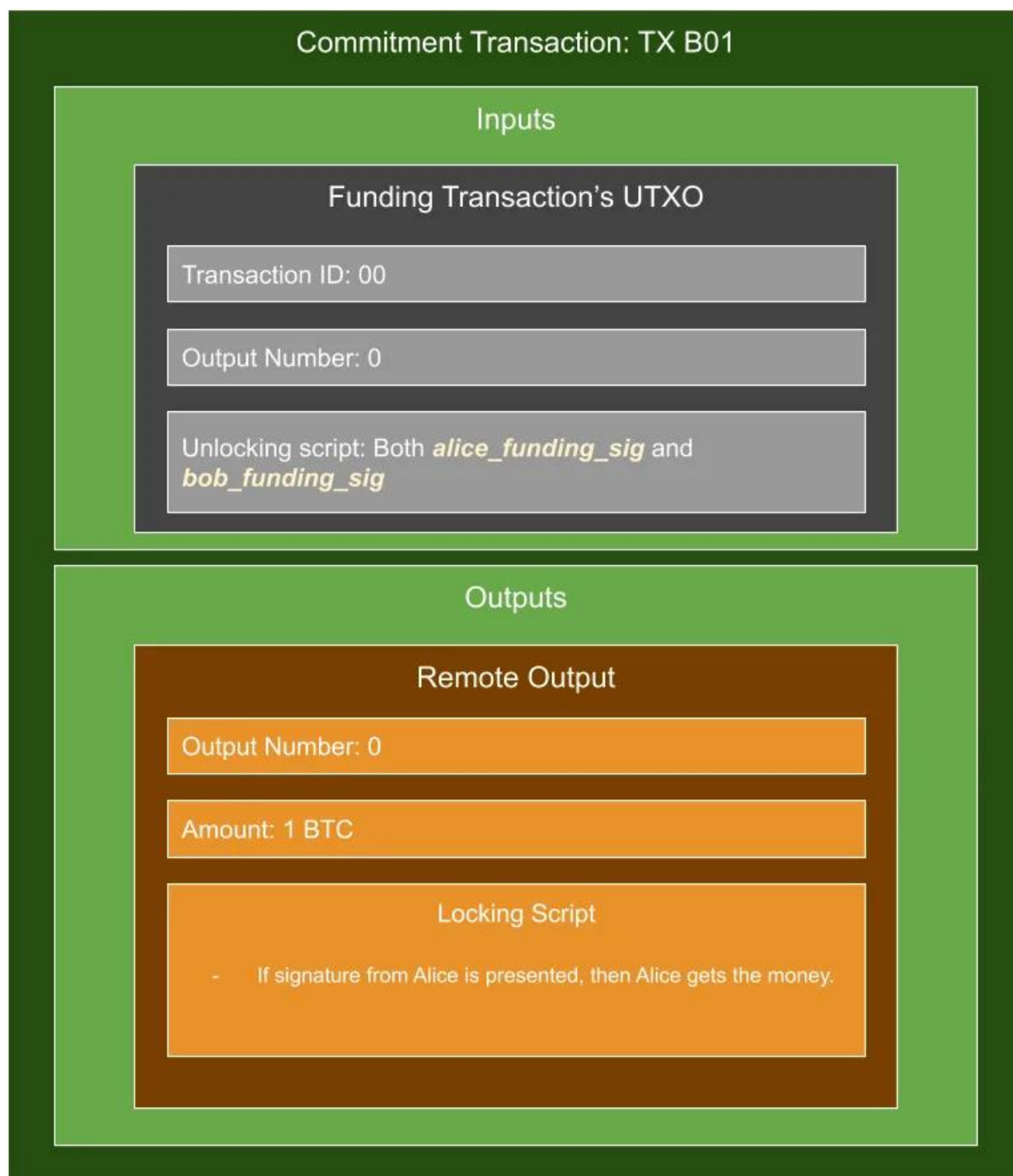


טרנזקציית התקצוב של אליס

שלישית, אליס ובוב יוצרים כל אחד את טרנזקציות ההתחייבות על בסיס טרנזקציית התקצוב, חותמים אותן ומחליפים ביניהן את החתימות.



טרנזקציית ההתחייבות של אליס. שימו לב שאין מאזן מרוחק (Remote Output).



טרנזקציית ההתחייבות של בוב. שימו לב שאין מאזן מקומי (Local Output).

לבסוף, אליס מוודאת את החתימות שקיבלה מבוב ומשדרת את טרנזקציית ההתחייבות.

אודות חלוקת השמות של המפתחות

אנו משתמשים בקיצורים בשמות של המפתחות. מפתחות ציבוריים נקראים `pubkey` (במקום `public_key`), מפתחות פרטיים נקראים `prikey` (במקום `private_key`), וחתימות נקראות `sig` (במקום `signature`). לדוגמא `alice_funding_sig` היא החתימה שבה משתמשים כדי לשחרר את הנעילה של טרנזקציית התקצוב של אליס.

בכל פעם שאנו משתמשים ב־`revocation` בשם של מפתח, המשמעות היא שהוא נוצר ע"י שילוב מפתחות של אליס ובוב. לדוגמא, `revocation_pubkey_tx_a1` נוצר ע"י `alice_pubkey_tx_a1` ו־`bob_pubkey_tx_a1`. דוגמא נוספת: `revocation_prikey_tx_a1` נוצר ע"י `alice_prikey_tx_a1` ו־`bob_prikey_tx_a1`.

בכל פעם שיש `tx` בשם של מפתח ביטול (`revocation key`), המשמעות היא שהוא נוצר כדי לאפשר ביטול של טרנזקציה ספציפית. לדוגמא `revocation_sig_tx_a1` נועדה לבטל אל הטרנזקציה של אליס `TX A01`. בינתיים, אין מנגנון ביטול בטרנזקציות של בוב (כי כל הכסף הולך לאליס), אבל בקרוב יהיו.

את החתימה `revocation_sig_tx_a1` אפשר לייצר ע"י `revocation_prikey_tx_a1`, ולבדוק שהיא תואמת למפתח הציבורי `revocation_pubkey_tx_a1`.

ברגע שמספר ה־`confirmations` של טרנזקציית התקצוב מגיע ל־`N`, הערוץ נפתח ונשאר פתוח. אליס מציעה את הערך `N`, ובוב יכול לסרב להקמת הערוץ אם הוא חושב ש־`N` גדול מדי. ל־`Output` מרוחק (`Remote Output`) מתייחסים באופן שונה מאשר ל־`Output` מקומי (`Local Output`), מכיוון שהמשתתף המקומי: א. לא יכולה לבזבז את ה־`Output` המרוחק ו־ב. אין שום אינטרס לשדר אותו. לדוגמא, לבוב אין שום אינטרס לשדר את `TX B01` לעולם, מכיוון שהוא לא ירוויח מזה כלום, ולכן אין צורך לשלם בטרנזקציה מפתח ביטול (`revocation_sig_tx_b1`). מאידך, ל־`Output` מקומי חייבות להיות גם `Timeout` וגם יכולת ביטול, מכיוון שלמשתתף המקומי יש אינטרס לרצות לקבל את כספו חזרה אם הצד השני לא מגיב, וגם אינטרס לנסות לרמות אם המאזן שלה ירד בעתיד.

עדכון מאזנים בערוצים - טרנזקציית ההתחייבות השנייה, ואלה שאחריה.

התהליך המורכב ביותר מתרחש כאשר מעדכנים את המאזן בערוץ. כאשר המאזן בערוץ משתנה, זה בא לידי ביטוי בטרנזקציות ההתחייבות החדשות. הכסף שנשלח/מתקבל נרשם ב־`Output`ים של ה־זכות/חובה (`Credit/Debit`), והצדדים מחליפים ביניהם את מפתחות הביטול של הטרנזקציות הקודמות.

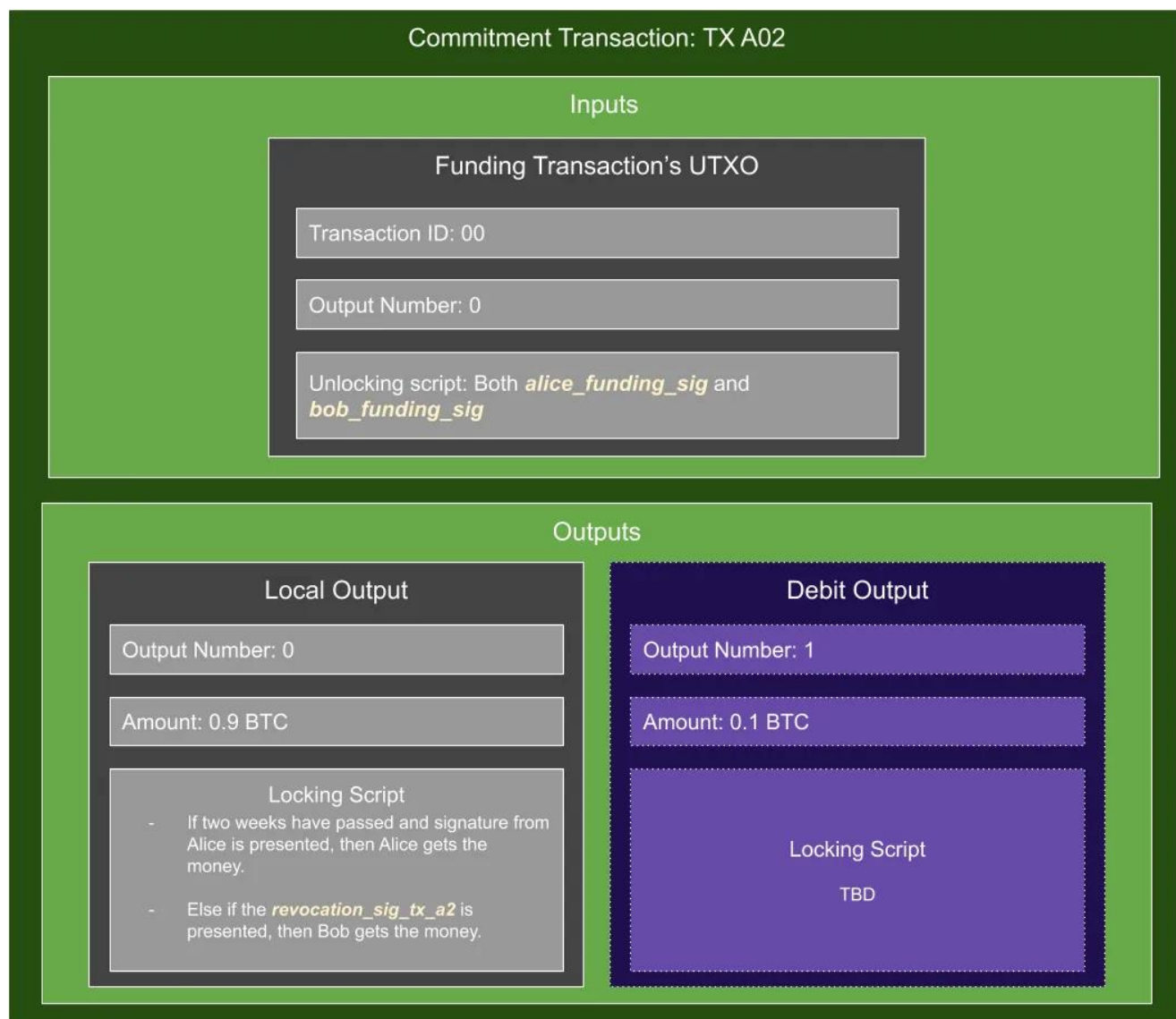
לדוגמא, אם אליס שולחת לבוב 0.1 ביטקוין, שציהק ייצרו טרנזקציות חדשות ויחליפו ביניהן את המפתחות. ראשית, הן יחליפו ביניהן את המפתחות הפרטיים `alice_prikey_tx_a1` ו־`bob_prikey_tx_b1`. כעת, כשלוב יש את `alice_prikey_tx_a1`, הוא יכול לחשב את

revocation_prikey_tx_a1, ולייצר את revocation_sig_tx_a1, אשר משחררת את הנעילה של ה- Output המקומי ב-TX Ao1 (במידה ואליס תשדר אותה). שנית, הק מחליפה את מפתחות הביטול הציבוריים החדשים, bob_pubkey_tx_a2, alice_pubkey_tx_b2, alice_pubkey_tx_a2, ו- bob_pubkey_tx_b2, אשר מייצרים את revocation_pubkey_tx_a2 ו- revocation_pubkey_tx_b2, ומשתמשות בהם בטרנזקציות ההתחייבות החדשות.

(הערת תרגום: לדעתי הכותב התבלבל כאן בסדר הכרונולוגי - קודם המשתתפים ישתפו את מפתחות הביטול הציבוריים החדשים, ואת החתימות של טרנזקציות ההתחייבות החדשות, ורק אחרי זה הק יחליפו ביניהם את המפתחות הפרטיים של הטרנזקציות הישנות, כדי להבטיח לצד השני שהק לא ירמו - אם הן ישודרו הצד השני יכול להגיב בזמן ולקחת את כל הכסף).

בטרנזקציית ההתחייבות החדשה של אליס, TX Ao2, יש שני Outputים:

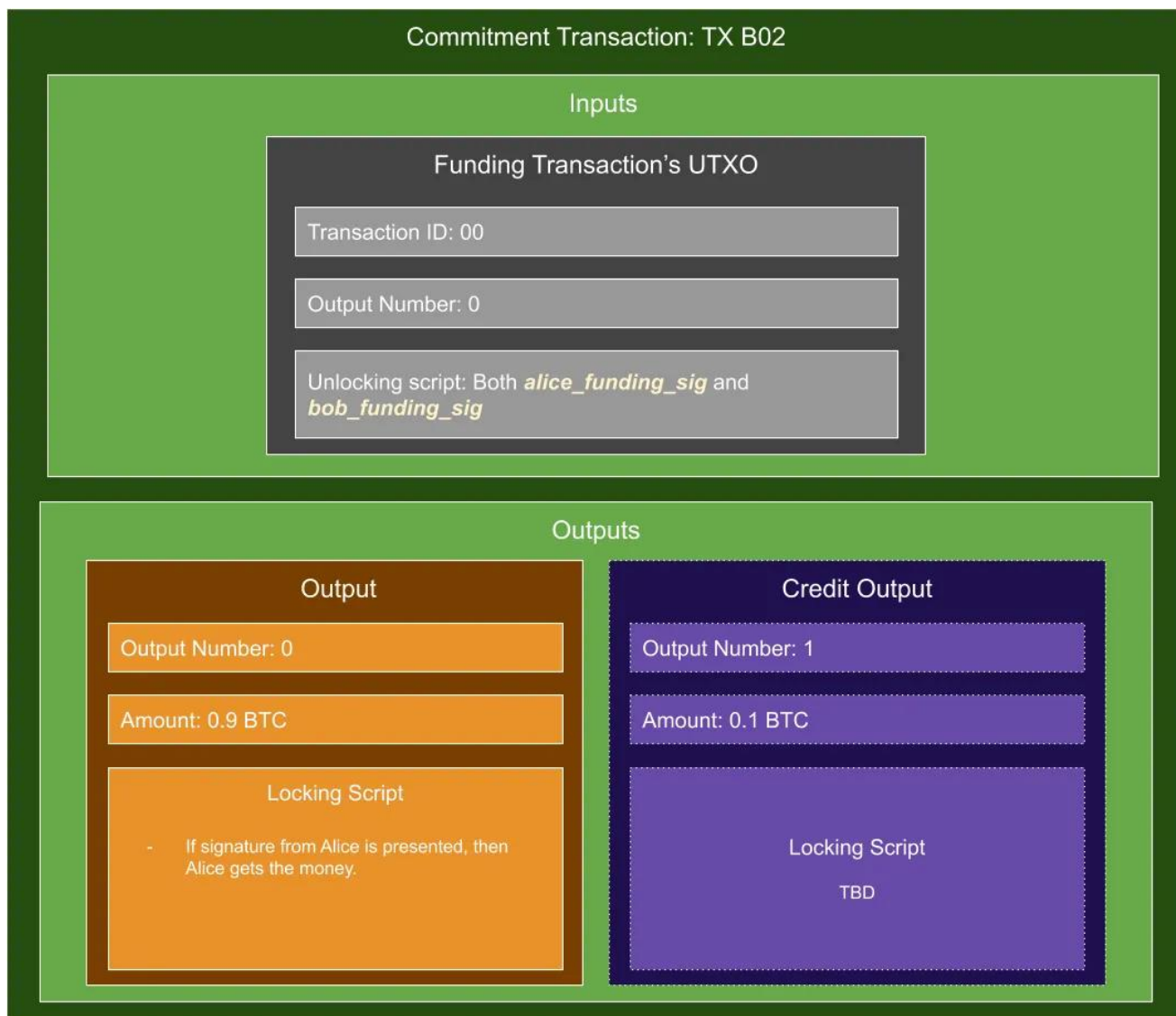
- Output מקומי, ששומר את המאזן הנוכחי שלה: 0.9 ביטקוין.
- Output חובה, ששומר את הכסף שהיא שלחה לבוב: 0.1 ביטקוין.



טרנזקציית ההתחייבות השנייה של אליס. שימו לב שכל טרנזקציות ההתחייבות תמיד מצביעות לאותה טרנזקציית תקצוב ראשונית.

מהצד של בוב, הוא מחזיק ב-TX B02, עם ה-Outputים:

- Output מרוחק, ששומר את המאזן של אליס: 0.9 ביטקוין.
- Output זכות, ששומר את הכסף שהוא מקבל: 0.1 ביטקוין.



טרנזקציית ההתחייבות השנייה של בוב. שימו לב שכל טרנזקציות ההתחייבות תמיד מצביעות לאותה טרנזקציית תקצוב ראשונית.

לפני שניכנס לתכנון של ה-Outputים החדשים (חובה/זכות), בואו נוודא שה-Output המקומי והמרוחק אכן תקפים ולא ניתן לרמות.

Outputים מקומיים ומרוחקים מאובטחים

ברור מראש שלבוב אין שום אינטרס לשדר את הטרנזקציה הישנה TX B01, כי ב-TX B02 הוא יקבל יותר כסף. מצד שני, עבור אליס:

- אם בוב נעלם, היא יכולה לשדר את TX A02 ואז אחרי שבועיים היא תוכל להשתמש בכסף.
- בנוסף, אם אליס משדרת את TX A02, לבוב אין דרך לקחת את ה-Output המקומי, מכיוון שאין לו את `revocation-prikey-tx-a2`, שנוצר ע"י `alice-prikey-tx-a2` ו-`bob-prikey-tx-a2` (לבוב יש את השני אבל לא את הראשון).

- אם אליס משדרת את טרנזקציית TX A01 הישנה, היא צריכה לחכות שבועיים לפני שהיא יכולה להשתמש בכסף. באותו זמן, בוב יכול להשתמש ב-`alice_prikey_tx_a1` (שאליס נתנה לו) וב-`bob_prikey_tx_a1`, כדי ליצור את `revocation_prikey_tx_a1` ולייצר `revocation_sig_tx_a1` שיסחרר את ה-Output, מה שייתן לבוב את כל הכסף, כעונש לאליס. לפיכך, עדכון ה-Output המקומי וה-Output המרוחק הוא בטוח והוגן. מקבלת הכסף לא דואגת ששולח הכסף ינסה לרמות, ושולח הכסף לא דואגת שמקבלת הכסף יעלם באמצע. השאלה המסובכת יותר היא איך אנחנו מאבטחים את ה-Output של הזכות והחובה?

ביצוע תשלומים עם אפשרות ביטול

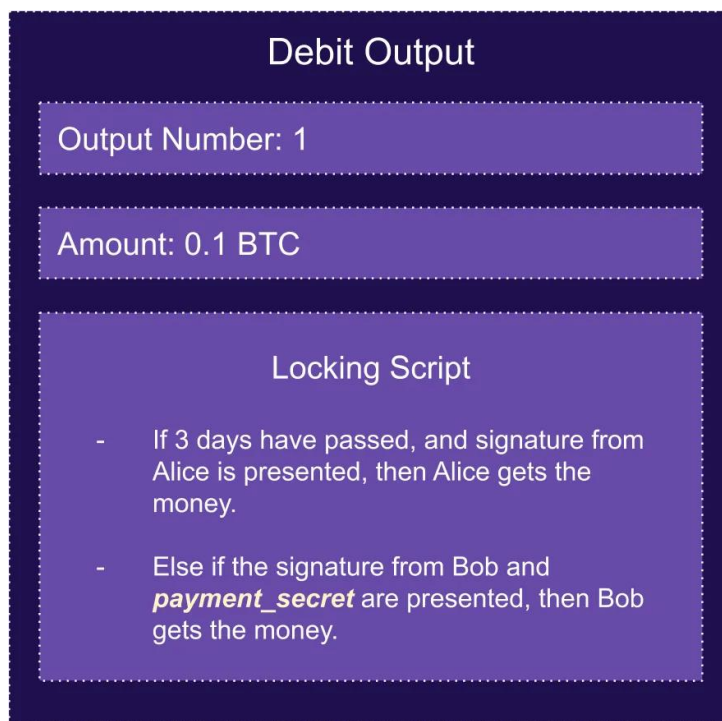
נניח שלבוב יש תרנגולת והוא מוכן למכור אותה לאליס עבור 0.1 ביטקוין ([למה? כי תרנגולת היא חברתו הטובה ביותר של האדם!](#)). הק לא ממהרזת ומצידק שייקח לתשלום אפילו שלושה ימים להגיע. בוב מתחיל את התהליך באמצעות "בקשת תשלום" (payment request). התהליך הוא כדלקמן:

1. בוב מייצר סוד בשם `payment_secret` ושומר אותו לעצמו (נקרא גם Preimage).
2. בוב מפעיל את פונקציית הגיבוב על `payment_secret`, כדי לייצר `payment_hash`, ושולח אותו לאליס.
3. אליס מייצרת טרנזקציית התחייבות חדשה, שבה יש Output חובה שאותו ניתן לבזבז באחד משני מקרים:

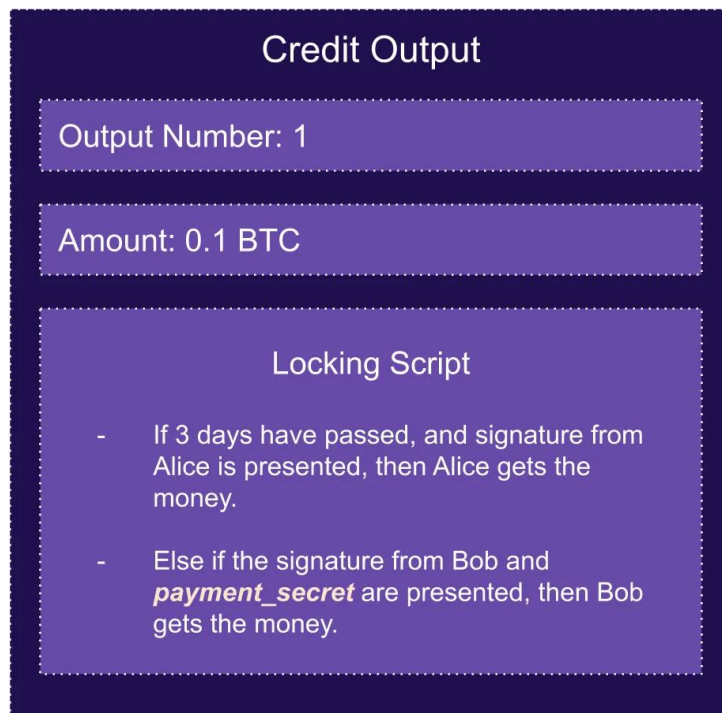
- אם בוב יכול לחשוף את ה-`payment_secret` בתוך שלושה ימים, בוב מקבל את הכסף.
- אם חלפו שלושה ימים מרגע ההסכם ובוב לא חשף את ה-`payment_secret`, אליס מקבלת את הכסף שלה חזרה.

4. בוב מייצר טרנזקציית התחייבות חדשה דומה, שבה במקום Output חובה לאליס יש Output זכות לבוב, תחת אותם תנאים.

נניח שה-Outputים החדשים ייראו כך:



Output החובה של אליס ב-TX Ao2. האם הוא יעבוד בכל המקרים והתגובות? הערת מתרגם: ה-Else כאן הוא לא במובן של שפת-תכנות, כלומר הוא לא מבצע בדיקה ש-"חלפו פחות משלושה ימים". למעשה אין אופרטור בסקריפט של ביטקוין שבודק דבר כזה (הסבר בהמשך). המשמעות של ה-Else כאן היא "עוד דרך לבזבז את ה-Output, בין אם חלפו או לא חלפו שלושה ימים".



Output הזכות של בוב ב-TX Bo2, שכרגע זהה ל-Output חובה של אליס. האם הוא יעבוד בכל המקרים והתגובות? הערת מתרגם: ההערה הקודמת על ה-Else תקפה גם פה.

המטרה שלנו היא ליצור מנגנון שבו אם בוב יודע ומוכן לחשוף את `payment_secret` בזמן הקצוב, בוב יקבל את הכסף במאזן החדש, ואם אליס לא מגיבה הוא יכול לשדר לרשת הביטקוין טרנזקציה (שכוללת את ה-`payment_secret`) שתיתן לו את הכסף. כל מי ששופט מהצד ומסתכל על הבלוקצ'יין יראה שבווב אכן ידע את ה-`payment_secret` בזמן הקצוב. מאידך, אם הוא לא חשף לאליס/לבלוקצ'יין את ה-`payment_secret` בזמן הקצוב, לא מגיע לו לקבל את הכסף, אפילו אם הוא פתאום התעורר והתחיל לשדר טרנזקציות כלשהן. אם בוב לא מצליח לחשוף לאליס את `payment_secret` בזמן הקצוב, ומפסיק להגיב כדי שיחזרו למאזן הישן, אליס צריכה להיות מסוגלת לשדר בסוף הזמן הקצוב טרנזקציה שתחזיר לה את הכסף. האם לדעתכם ה-`Output`ים האחרונים מקיימים את כל הדרישות של כל המקרים והתגובות הללו?

זה מבלבל, אני יודע. מכיוון שהדרישה של רשת הברק היא לדאוג שהטרנזקציות יגיעו לרשת הביטקוין לעיתים רחוקות ככל הניתן, רק כאשר אחד הצדדים מפסיק להגיב או פועל בזדון. במצב הרגיל, אנחנו רוצים לתכנן דרך שבה אליס ובוב ירגישו בטוחים לגבי הכסף שלהם, מבלי לשדר את טרנזקציות ההתחייבות, ושהערוץ יישאר פתוח ככל שניתן. במקרה שלנו, בוב כבר יודע את ה-`payment_secret` בתוך שלושה ימים, אבל אנחנו לא רוצים לחייב אותו לשדר את הטרנזקציה כדי להוכיח את זה. הערת מתרגם: בדוגמאות הנ"ל, הזמן הקצוב הוא יחסי ולא אבסולוטי ("תוך 3 ימים מרגע שליחת הטרנזקציה"), כלומר אם אליס רוצה להפעיל לחץ על בוב לחשוף את `payment_secret`, היא חייבת לשדר את הטרנזקציה - ואנחנו רוצים להימנע מזה. אבל אפילו אם נמצא דרך כזו, אין מנגנון ביטול לטרנזקציה, כלומר אם המשתתפים יחליטו על מאזן חדש בעתיד, שום דבר לא יעניש את מי שינסה לשדר את הטרנזקציה הישנה הזו. אפילו אם בוב חשף בפני אליס את `payment_secret`, ואליס הסכימה על מאזן חדש בטרנזקציות התחייבות חדשות שבה יש לבוב יותר כסף ב-`Output` המקומי/מרוחק בהתאמה, בוב ידאג שאליס עלולה להשתמש באופן לא הוגן בטרנזקציה הישנה הזו (אחרי שלושה ימים אבסולטיים). הסבר מפורט יגיע בהמשך.

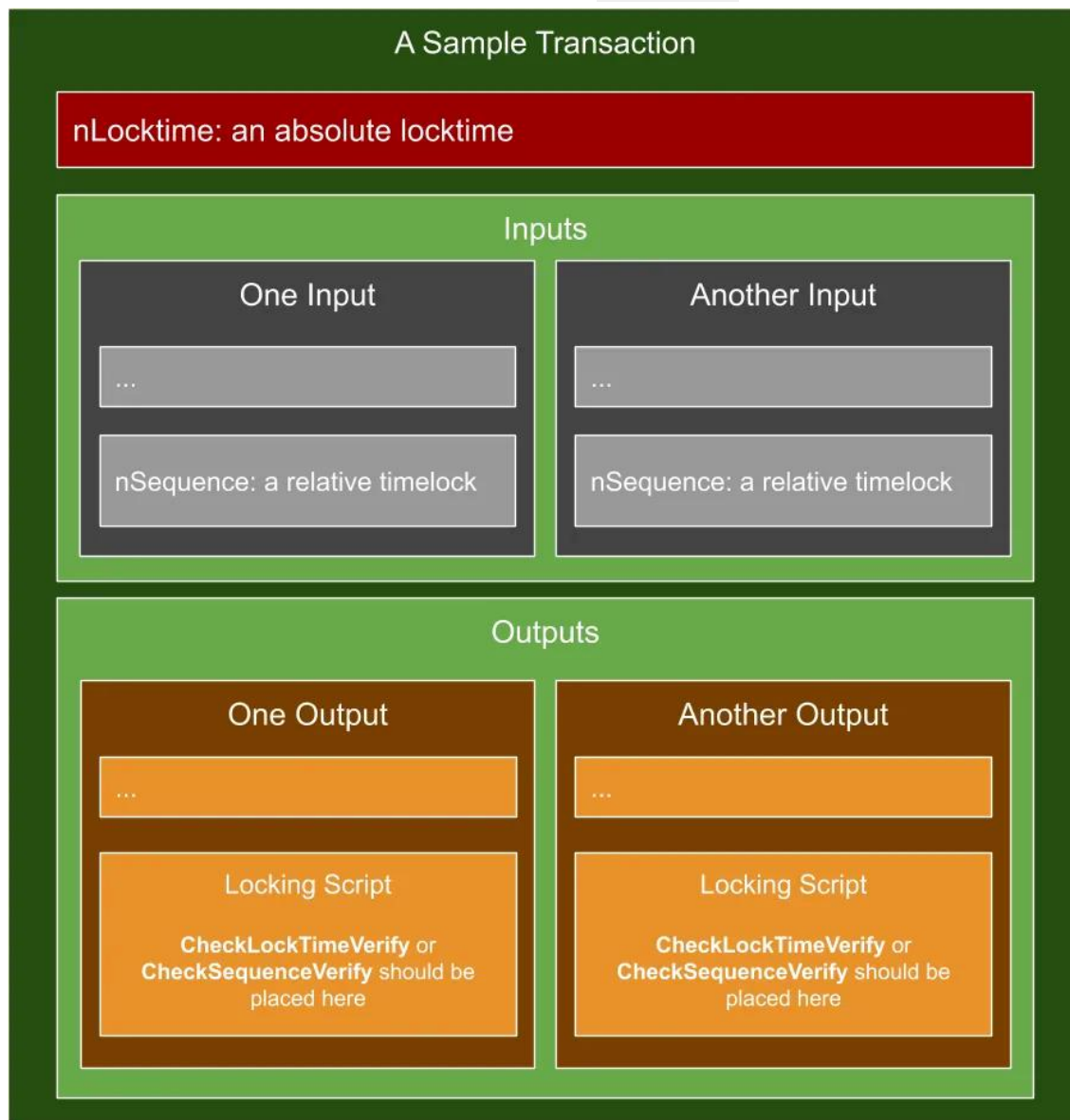
אז לפני שבווב שולח את התרנגולת האהובה שלו, אנחנו חייבים להבין לעומק את מנגנוני נעילת-הזמן, ואיך אנחנו יכולים להשתמש בהם.

נעילת זמן בביטקוין

בביטקוין יש שני סוגים של נעילות-זמן, אחד הוא זמן אבסולוטי, והשני זמן יחסי. ההבדל בינם הוא פשוט כמו "ניפגש בשעה 15:00" מול "ניפגש בעוד שעתיים" - כאשר נעילת הזמן היחסי מתחילה ברגע שהטרנזקציה נכרית בבלוק. המקום שבו שמית את נעילת הזמן קובע מתי אפשר לשדר את הטרנזקציה או מתי אפשר להשתמש/לבזבז את ה-`Output`ים שלה. בתוך כל טרנזקציה, יש שלושה מיקומים שקובעים את נעילת הזמן.

- בתוך מחלקת ה-`Input`, לכל `Input` יש שדה `nSequence` שמשתמשת בו עבור נעילת זמן יחסית.

- בתוך מחלקת ה-Output, לכל סקריפט נעילה של כל Output, האופרטור `CheckLockTimeVerify` (בקיצור CLTV) קובע את נעילת הזמן האבסולוטית, והאופרטור `CheckSequenceVerify` (בקיצור CSV) קובע את נעילת הזמן היחסית.
- ברמת הטרנזקציה, השדה `nLocktime` קובע נעילת זמן אבסולוטית.



טרנזקציה לדוגמא שבה מודגשים המיקומים של נעילות הזמן

רשת הברק משתמשת ב-`CheckSequenceVerify`, `nLocktime`, ו-`CheckLockTimeVerify` בתכנון שלה.

השדה `nLocktime` שולט מתי לטרנזקציה מותר להיכנס לבלוקצ'יין. אם יש טרנזקציה שבה כתוב `nLocktime` של 2014/01/01 12:00, ניתן יהיה לשדר אותה רק בעוד יותר ממאה שנים. האופרטורים `CheckLockTimeVerify` ו-`CheckSequenceVerify` שולטים במתי ניתן לבדוק את ה-Output. אם הסקריפט מכיל `CheckSequenceVerify` עם "30 ימים", ניתן יהיה לבדוק את ה-Output רק אחרי

30 ימים. אם הסקריפט מכיל `CheckLockTimeVerify` עם `2140/01/01 12:00`, ניתן יהיה לבזבז את ה-`Output`ים רק כשהגיע התאריך `2140/01/01 12:00`. באופן פשוט, `nLocktime` שולט בזמן השידור של הטרנזקציה, ו-`CheckLockTimeVerify`, `CheckSequenceVerify` שולטים בזמן השימוש ב-`Output`ים של הטרנזקציה.

הערת מתרגם: שימו לב שכל נעילות הזמן הן "מותר לבצע פעולה X החל מנקודת זמן Y", ואין אף פעולה בסגנון "מותר לבצע פעולה X רק עד נקודת זמן Y". זהו עקרון בסיסי חשוב ברשת הביטקוין: מרגע שיש טרנזקציה שניתן להכניס לבלוקצ'יין, תמיד יהיה אפשר להכניס אותה לבלוקצ'יין, אלא אם כן אחד ה-`Input`ים שלה מבוזבז (ניסיון `Double-Spend` או `Replace-By-Fee`). הסיבה היא להקל על הצמתים כשמגיע בלוק חדש. נניח שטרנזקציה ב-`Mempool` יכולה הייתה להיכנס לבלוק האחרון, כלומר הצומת בדק שמתקיימים שני התנאים:

- ה-`Input`ים שלה מצביעים על `Output`ים שלא בוזבזו עדיין (`UTXO`) ולא מתנגשים עם `Input`ים של טרנזקציה אחרת ב-`Mempool` (פרט למקרה `Replace-By-Fee`).
- אפשר להריץ בהצלחה את סקריפט הנעילה והשחרור של ה-`Input`ים שלה.

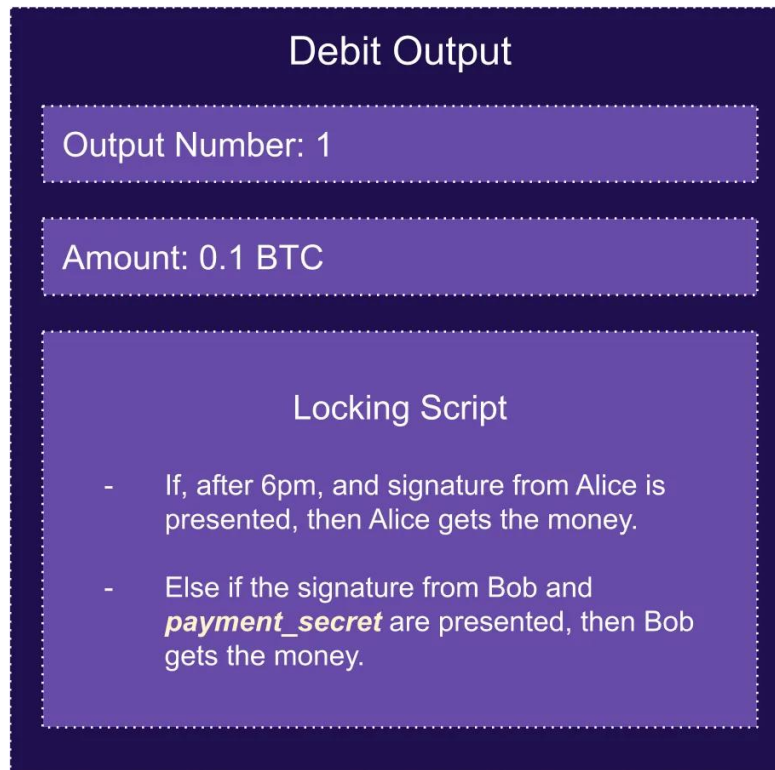
ונניח שעכשיו מגיע בלוק חדש - מספיק לנו לעבור על הטרנזקציות ב-`Mempool` ולבדוק את הסעיף הראשון (ה-`Input`ים לא מתנגשים עם ה-`Input`ים של הטרנזקציות בבלוק החדש), ואין צורך לבדוק שוב את כל סקריפטי הנעילה והשחרור. אם הסקריפטים רצו בעבר בהצלחה הם בטוח ירוצו בהצלחה גם עכשיו.

שימוש בנעילת זמן ב-`Output`ים

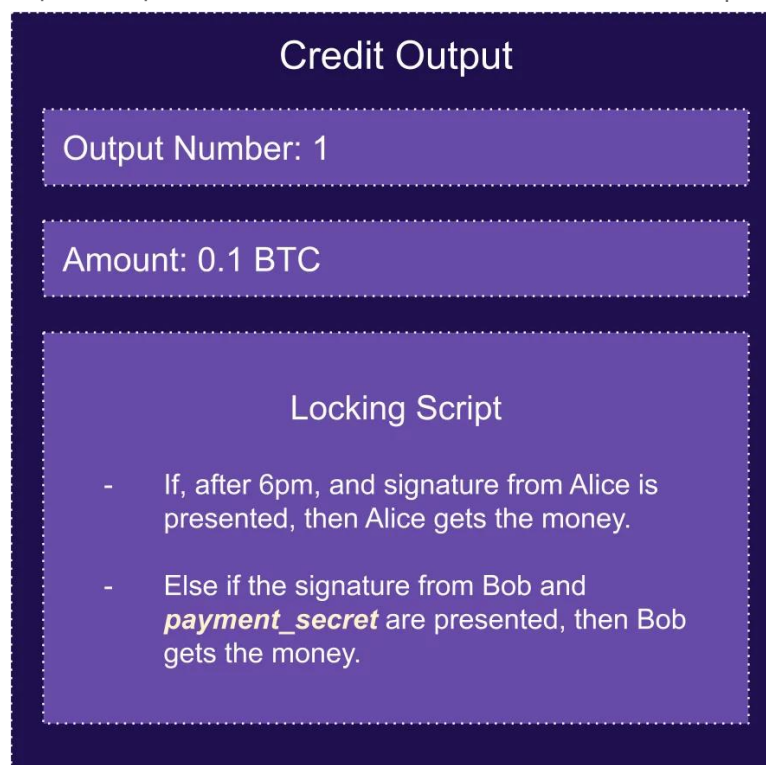
`Output`ים ניתנים לביטול, כמו ה-`Output` המקומי והמרוחק, משתמשים בנעילת-זמן יחסית, מכיוון שהם רלוונטיים כאשר נחשפת טרנזקציית התחייבות ישנה (ע"י משתתף רמאיז). לפיכך, הגיוני להתחיל את הספירה לאחר כאשר הטרנזקציה הישנה משודרת או נכרית. לדוגמא, אם אליס מנסה לרמות ולשדר את `TX A01`, ברגע שהטרנזקציה נכרית ע"י הבורזת, היא צריכה לחכות שבועיים כדי לבזבז את ה-`Output`ים שלה, מה שנותן לבוב זמן לשים לב לכך ולהגיב (אם הוא עוקב אחרי הבלוקצ'יין). מצד שני, אם נשתמש פה בנעילת זמן אבסולוטית, לדוגמא `2019/12/31 23:59:59`, אליס תוכל לחכות לערב ראש השנה האזרחית כדי לשדר את הטרנזקציה ומייד לבזבז את ה-`Output` שלה, מבלי שבוב שם לב (נניח כי חיבור האינטרנט שלו היה איטי באותו רגע).

`Output`ים לרישום תשלומים שמתבצעים כרגע, כמו `Output` חובה ו-`Output` זכות, משתמשים בנעילת-זמן אבסולוטית, מכיוון שהם נועדו להפעיל לחץ על מקבלת התשלום (במקרה שלנו בוב) לחשוף את הסוד, מבלי לשדר את הטרנזקציה. אם בוב לא יכול לספק את ה-`payment_secret` לפני זמן אבסולוטי מסוים, לדוגמא "היום ב-18:00", אליס צריכה לקבל חזרה את היכולת לבזבז את הכסף (אחרי 18:00). אם היינו משתמשים בזמן יחסי "שלושה ימים", לבוב לא היה שום אינטרס למהר, אלא אם כן הוא היה רואה שאליס ממש שידרה את הטרנזקציה.

נניח שהק מבצעת את התשלום ב-8:00 היום, ואליס רוצה לתת לבוב 10 שעות לחשוף את ה-
`payment_secret`. האם יספיק לשנות את ה-Outputים שייראו כמו בשרטוט הבא?



Output החובה של אליס ב-TX A02. ההערה על ה-Else תקפה גם כאן.

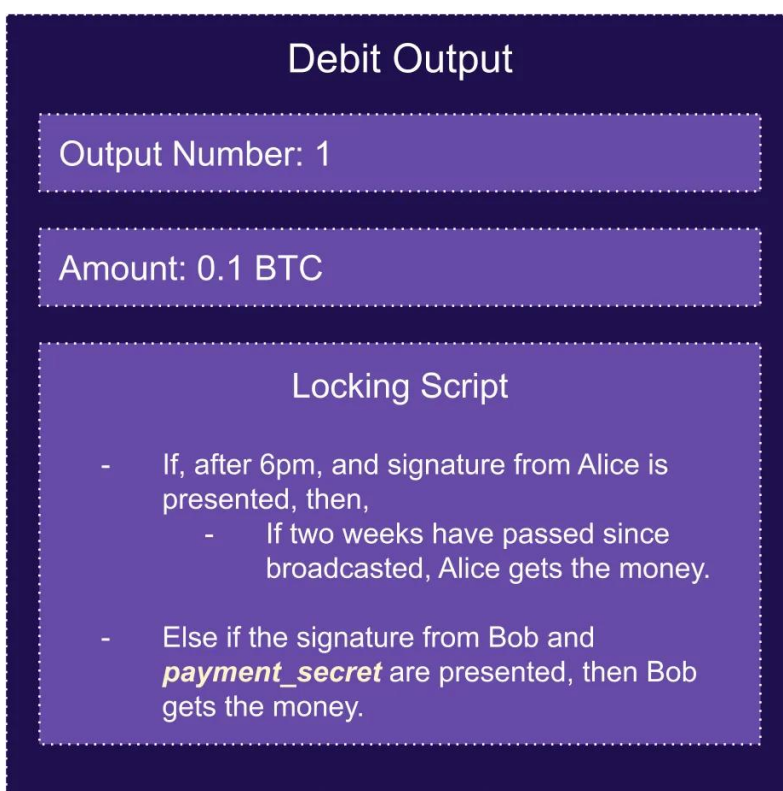


Output הזכות של בוב ב-TX B02, שברגע זה ל-Output חובה של אליס. ההערה על ה-Else תקפה גם כאן.

נעילת הזמן האבסולטית פתרה את הבעיה האחרונה, מכיוון שאף אחת לא צריכה לשדר אף טרנזקציה כדי להתחיל להפעיל את מנגנוני נעילת הזמן.

אבטחת Output החובה ו-Output הזכות

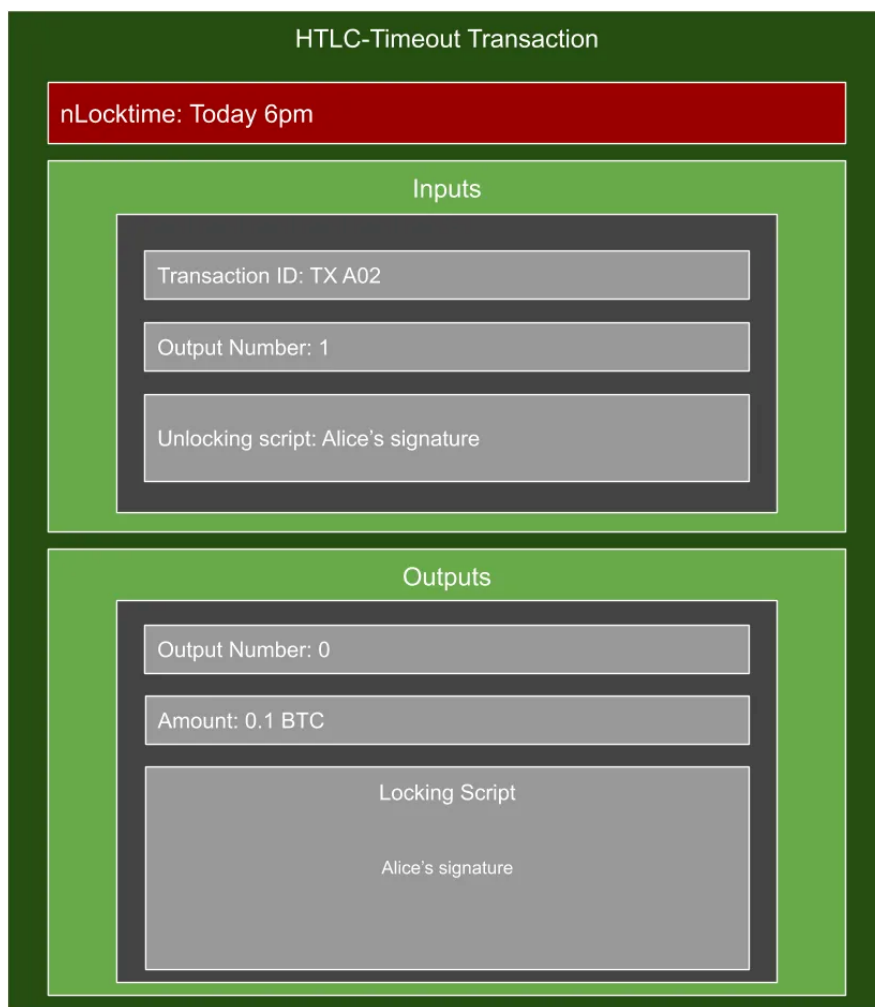
השעה 17:00. בעוד שעה אחת בלבד, אליס תוכל לשדר את TX Ao2 ולקחת לעצמה את הכספים ב-Output החובה. למרות שלבוב יש את ה-`payment_secret` והוא מוכן למסור אותו לאליס, נראה שאליס תמיד תוכל לשדר את TX Ao2 בעתיד ולקחת את הכסף, אפילו אם היא תיתן לו טרנזקציות התחייבות חדשות. הדרך היחידה של בוב למנוע את זה היא לשדר בעצמו את TX Bo2. האם יש דרך אחרת למנוע מאליס לרמות? אולי ננסה להוסיף נעילת זמן יחסית, לדוגמה שבועיים, ל-Output החובה של אליס.



Output החובה של אליס ב-TX Ao2. גם כאן המשמעות של ה-Else היא "עוד דרך לבזבז את ה-Output בין אם הגיעה או לא הגיעה השעה 18:00".

אם אליס הסכימה למאזנים חדשים בטרנזקציות התחייבות חדשות, אבל פתאום מרמה ומשדרת את TX Ao2 הישנה, היא צריכה לחכות שבועיים כדי לבזבז את הכסף שלה. באותו זמן, בוב יכול לגלות את זה, ולבזבז את ה-Output הזה באופן מיידי (עם ה-`payment_secret` שלו). אבל עכשיו תורה של אליס לדאוג! נניח הגיעה השעה 18:00 בלי שבוב הגיב ובלי שהוא חשף בפניה את ה-`payment_secret`. אם אליס רוצה את כספה חזרה ותשדר את TX Ao2, בוב יכול לחכות שבוע שלם עם ה-`payment_secret` שמור אצלו בסוד, ורק אז לחשוף את `payment_secret` ולבזבז את ה-

Output. בוב יכול לרמות ולא לעמוד בהסכם (לפרסם את payment_secret לפני 18:00), ועדיין לקבל את הכסף. האם יש דרך למנוע מבוב לעשות את זה? במקום לשים נעילת-זמן על Output החובה של אליס, נאפשר לה לבזבז את ה-Output הזה באופן מיידי אבל עם תנאי אחד: אליס תבטיח לבזבז אותו ל-input של טרנזקציה מיוחדת חדשה, שנקראת HTLC-Timeout.

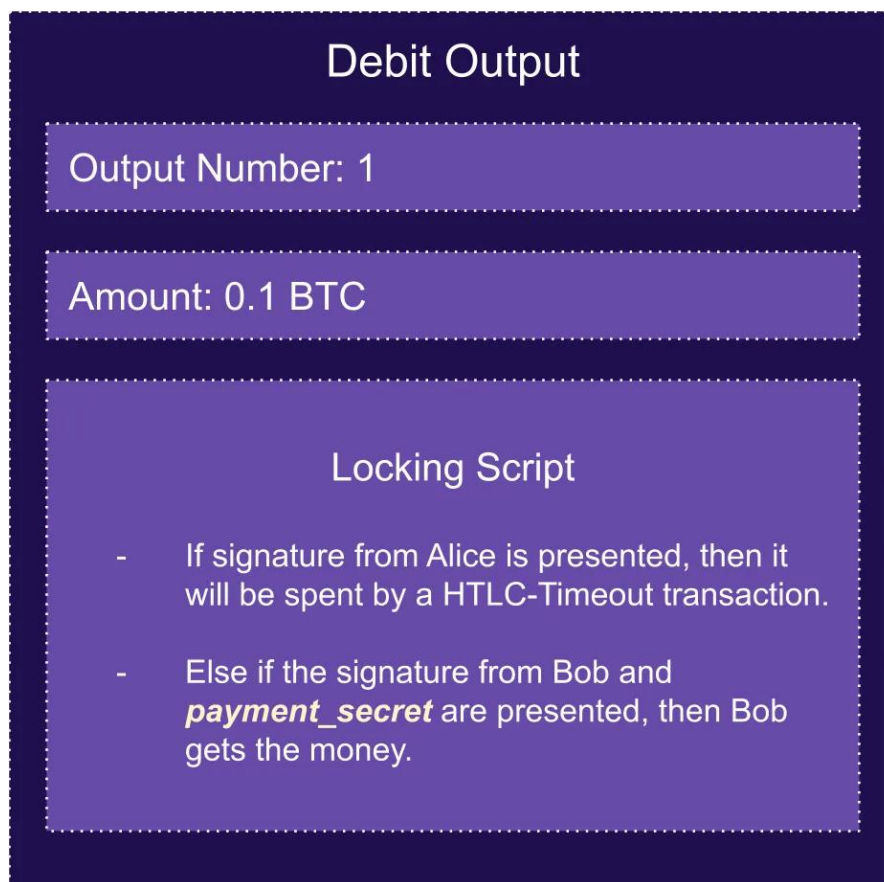


טרנזקציית ה-HTLC-Timeout של אליס, שאת ה-Output ים שלה רק אליס יכולה לבזבז.

טרנזקציית ה-HTLC-Timeout משתמשת בנעילת-זמן אבסולוטית (nLocktime),

- היא משתמשת ב-Output מספר 1 של TX A02 (שהוא Output החובה של אליס).
- יש לה nLocktime של 18:00, כלומר לא ניתן לשדר אותה לפני השעה 18:00.

גם Output החובה עצמו צריך להשתנות:



Output החובה של אליס ב-TX Ao2

הבעיה שלנו כמעט נפתרה. מהצד של אליס, אם הגיעה השעה 18:00 ובוב לא חשף את ה-**payment_secret**, היא עושה שתי פעולות:

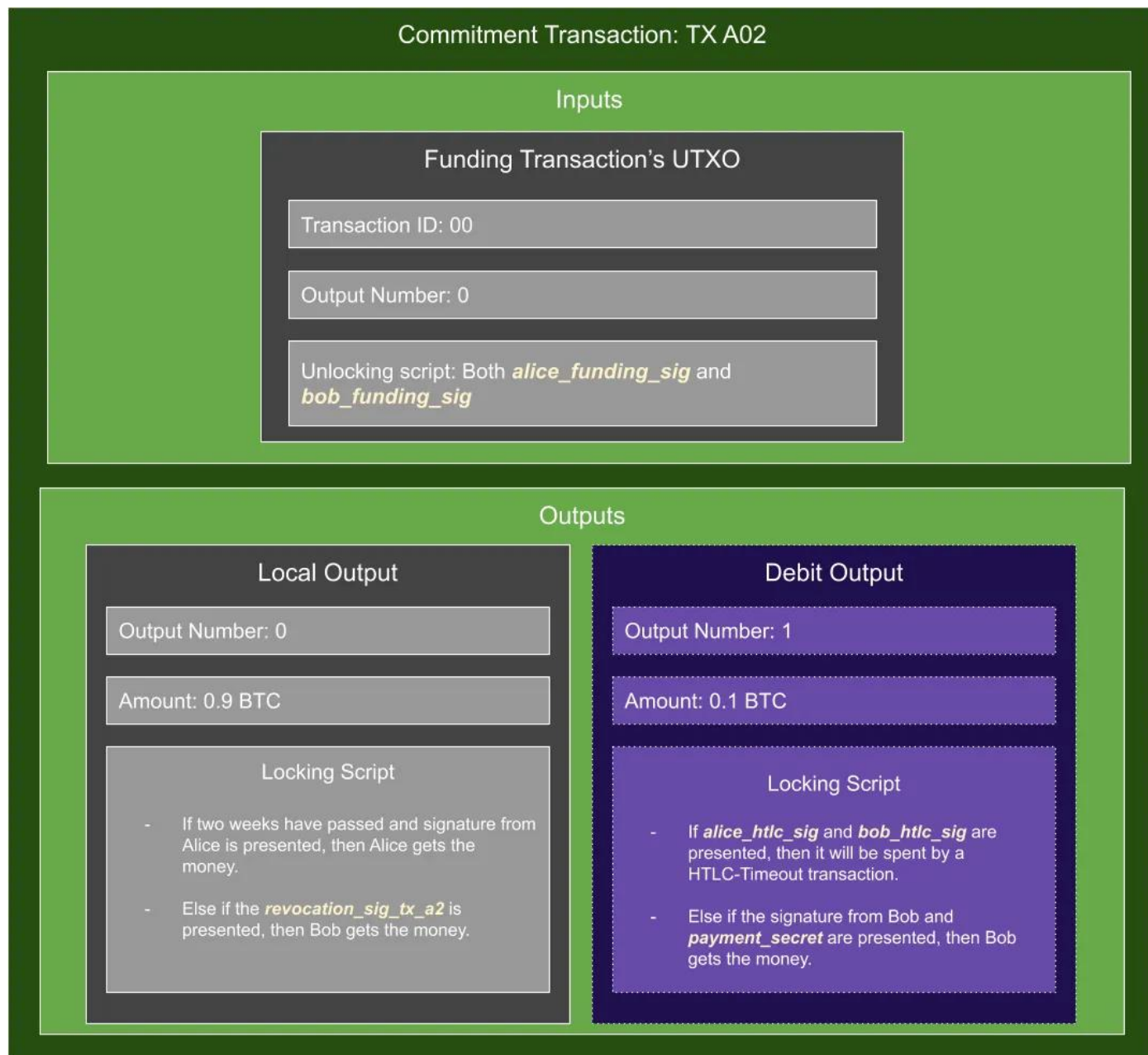
1. היא משדרת את TX Ao2 תחילה.

2. היא משדרת את HTLC-Timeout שמצביעה ל-TX Ao2.

לפני השעה 18:00, אין לאליס שום אינטרס לשדר את TX Ao2, מכיוון שאת ה-Output הזה היא אמורה לבזבז רק עם HTLC-Timeout (ואת HTLC-Timeout ניתן לשדר רק בשעה 18:00 בגלל ה-nLocktime). כשהגיעה השעה 18:00, אפילו אם בוב מוכן לחשוף את ה-**payment_secret**, הוא לא יכול להשתמש ב-Output החובה הזה מכיוון שהוא כבר בוזבז ע"י ה-HTLC-Timeout שאליס שידרה (זה ייראה ברשת כמו ניסיון Double-Spend). למרות שסקריפט הנעילה מאפשר לבזבז את ה-Output בשתי דרכים שונות, רשת הביטקוין מוודאת עבורנו ש-Output יתבזבז לכל היותר פעם אחת בלבד. הערת מתרגם: בנוסף, סקריפט הנעילה ב-Output של ה-HTLC-Timeout יכול להתנהג דומה לזה של ה-Output המקומי/המרוחק - לאפשר לאליס לבזבז את הכסף אחרי שבועיים, ואם בוב שם לב לטרנזקציה ואליס נתנה לו את המפתח המתאים (בהבטחה שהיא לא תרמה), הוא יוכל לקחת את הכסף לעצמו מיד.

נראה שכמעט סיימנו, אבל פספסנו נקודה חשובה. אליס הבטיחה שבשעה 18:00 היא תבזבז את ה-Output החובה לתוך ה-HTLC-Timeout ולא לאף מקום אחר, אבל למה לה לעשות את זה?

היא לא תעשה את זה, אלא אם נוסף שינוי אחרון שבאמצעותו בוב יכריח אותה לעשות את זה. Output החובה לא יגיע לידיים של אליס ישירות, אלא לריבוי-חתימות שדורש גם את החתימה של אליס וגם את החתימה של בוב – ובוב יחתום מראש שהוא מסכים שה-Output במקרה הזה יבוצע רק ע"י ה-HTLC-Timeout.

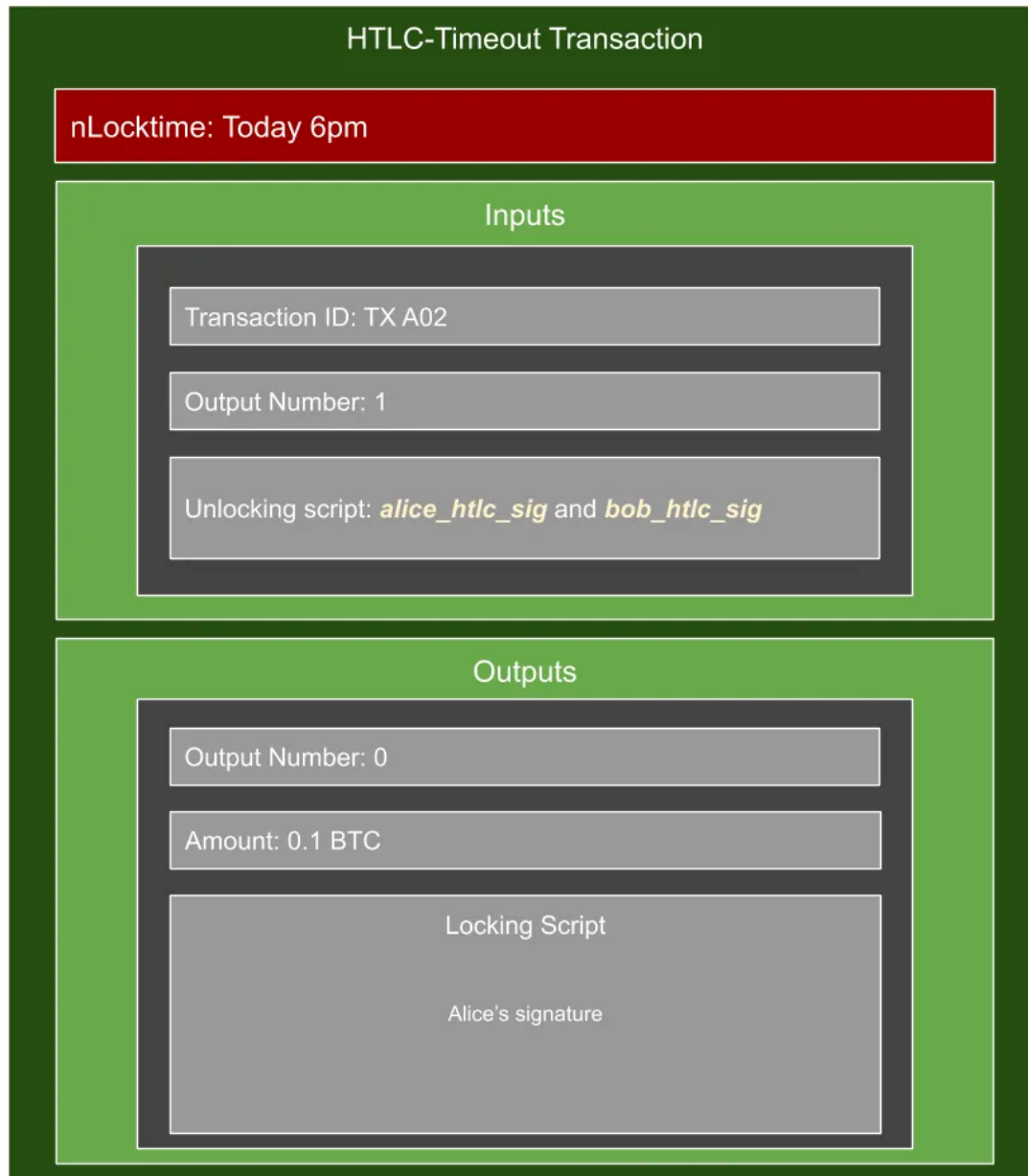


טרנזקציית ההתחייבות של אליס TX A02. שימו לב ל-multi-sig ב-Output החובה, שדורש גם את *alice_htlc_sig* וגם את *bob_htlc_sig*.

כשאליס יוצרת את TX A02, היא מבוצת את ה-Output של טרנזקציית התקצוב, מה שדורש גם את החתימה של אליס וגם את החתימה של בוב (*alice_funding_sig*, *bob_funding_sig*). לכן בוב יכול לבדוק ש-TX A02, נוצרה לפי הדרישות לפני שהוא שולח את החתימה שלו *bob_funding_sig* לאליס

(ואם אליס רוצה לשדר את הטרנזקציה, היא תצטרף את החתימה שלה `alice_funding_sig` ואז תשדר אותה).

טרנזקציית ה-HTLC-Timeout החדשה תיראה כך:



טרנזקציית ה-HTLC-Timeout של אליס. הערת מתרגם: בהמשך נסביר בפירוט מה מכיל סקריפט הנעילה (ה Locking Script).

כשיוצרות את טרנזקציית ה-HTLC-Timeout, היא מבזבזת את Output החובה מ-TX A02, מה שידרוש גם את החתימה של אליס וגם את החתימה של בוב (`alice_htlc_sig`, `bob_htlc_sig`). בוב אז בודק ש-HTLC-Timeout נוצרה באופן המתוכנן, ואז שולח את חתימתו `bob_htlc_sig` לאליס. בדרך זו, אליס תתנהג כמצופה ממנה, ואם הגיעה השעה 18:00 היא תוכל לבזבז את ה-Output רק לתוך טרנזקציית ה-HTLC-Timeout ולא למקומות אחרים. באופן כללי, אנחנו רוצים שהמשתתפים ירגישו

בטוחות לגבי הכסף שלהם, מבלי לשדר את הטרנזקציות, ולכן הושקע מאמץ גדול כדי לפתח את המנגנון המורכב הזה. עבור טרנזקציית TX Ao2 וטרנזקציית ה-HTLC-Timeout, המנגנון עובד כך:

- אם לבוב יש את payment_secret, הוא יודע בוודאות שהוא יכול לקבל את כספו לפני 18:00, כי אפילו אם אליס תשדר את TX Ao2, היא לא תוכל לבזבז את ה-Output עד השעה 18:00, ובוב ישתמש מייד ב-payment_secret כדי לקחת את כספו.
- אחרי השעה 18:00, אם בוב לא מגיב, אליס תוכל לשדר גם את TX Ao2 וגם את טרנזקציית ה-HTLC-Timeout כדי לקבל את כספה.

לפי התכנון הזה, לאליס אין אינטרס לשדר את TX Ao2 אלא אם כן היא חייבת. מהצד של בוב, ל-TX Bo2 יהיה מבנה דומה, ותהיה לו טרנזקציית HTLC-Success דומה מאוד לטרנזקציית ה-HTLC-Timeout.

Commitment Transaction: TX B02

Inputs

Funding Transaction's UTXO

Transaction ID: 00

Output Number: 0

Unlocking script: Both *alice_funding_sig* and *bob_funding_sig*

Outputs

Remote Output

Output Number: 0

Amount: 0.9 BTC

Locking Script

- If signature from Alice is presented, then Alice gets the money.

Credit Output

Output Number: 1

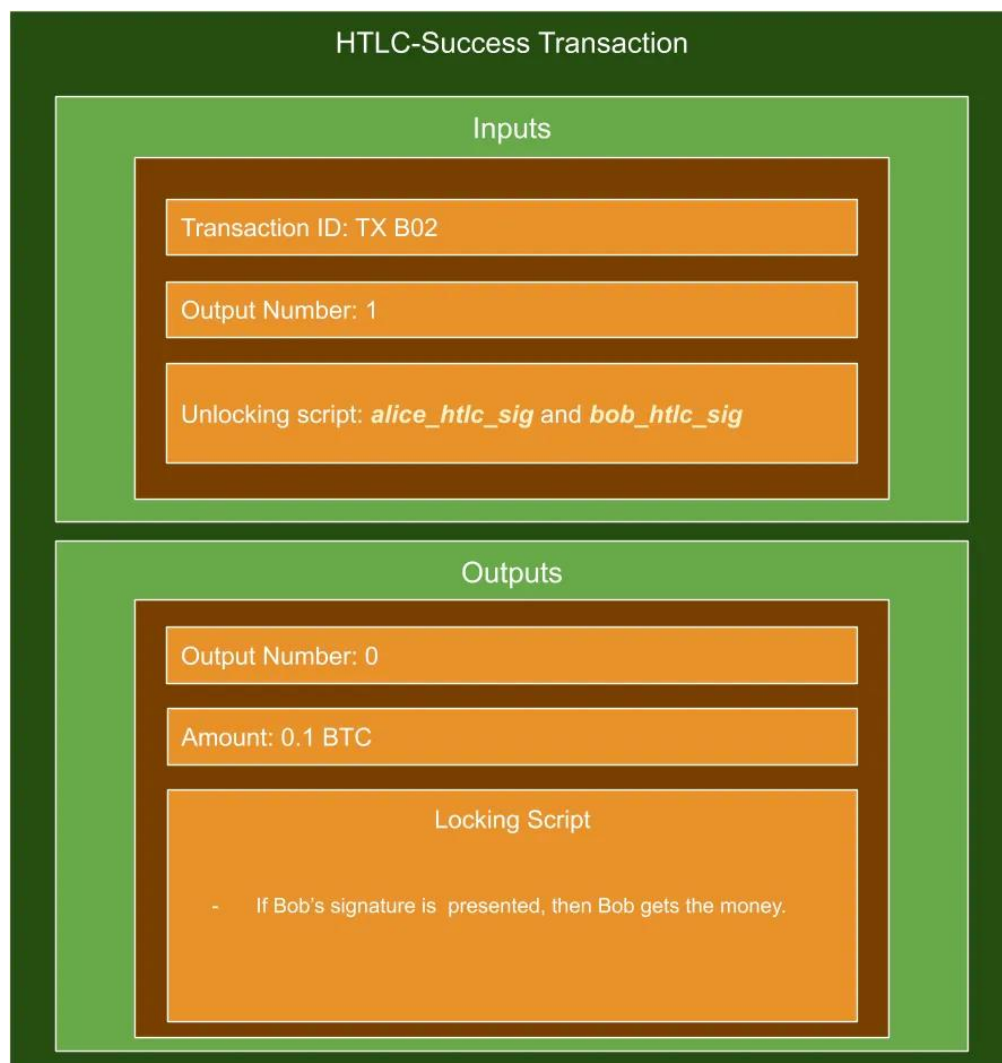
Amount: 0.1 BTC

Locking Script

- If *alice_htlc_sig*, *bob_htlc_sig* and *payment_secret* are presented, then it will be spent by a HTLC-Success transaction.
- Else If, after 6pm, and signature from Alice is presented, then Alice gets the money.

טרנזקציית התחייבות TX B02 של בוב. שימו לב ל-multi-sig ב-Output החובה, שדורש גם את *alice_htlc_sig* וגם את *bob_htlc_sig*.

ניתן לבדוק את Output הזכות של בוב באמצעות *alice_htlc_sig*, *bob_htlc_sig* ו-*payment_secret*. ביחד, ואז הוא יגיע לטרנזקציית ה-HTLC-Success. בדומה לטרנזקציית ה-HTLC-Timeout שנבדקת ע"י בוב לפני שהוא שולח לאליס את החתימה שלו, טרנזקציית ה-HTLC-Success נבדקת ע"י אליס לפני שהיא שולחת את החתימה שלה לבוב - כדי שבוב לא ינסה לעשות דברים חריגים.



טרנזקציית ה-HTLC-Success של בוב. הערת מתרגם: סקריפט השחרור (ה-Unlocking script) צריך לכלול גם את ה-`payment_secret` כדי להתאים ל-Output של **TX B02**. בנוסף סקריפט הנעילה (ה-Locking script) יראה קצת שונה: בוב לא אמור לקבל את הכסף מייד אלא כעבור שבועיים, שבהן יש לאליס זמן להגיב ולקחת לו את הכסף אם בוב מרמה והטרנזקציה כבר בוטלה (ע"י מפתח פרטי מסוים שבוטל ישלח לה).

אם בוב שולח לאליס את `payment_secret` לפני השעה 18:00 אבל היא לא מגיבה, ולא מוכנה לשלוח לו טרנזקציות התחייבות חדשות ואת המפתחות הדרושים כדי לבטל את הישנות, בוב יפרסם גם את **TX B02** וגם את טרנזקציית ה-HTLC-Success (שיכללו על הבלוקצ'יין את ה-`payment_secret` - סוג של הוכחה לכולם שבוטל אכן היה מוכן לפרסם את ה-`payment_secret` לפני השעה 18:00).

עדכון Output החובה ו-Output הזכות

היכולת לרשום עדכוני מאזן עם נעילות-זמן, היא לא מספיקה - עלינו להיות מסוגלים גם לעדכן אותם. נחזור לעסקת מכירת התרנגולת שהייתה לנו: השעה 18:00 תגיע רק בעוד מספר שעות ובוב עדיין לא סיפר לאליס את ה-`payment_secret` - אבל אליס עכשיו נזכרת שהיא רוצה לקנות מבוב גם כמה ביצים בנוסף לתרנגולת ומוכנה לשלם לו 0.05 ביטקוין נוספים. כדי לבצע את העסקה הנוספת, אליס

ובוב יצטרכו לייצר TX A03 ו-TX B03 חדשים, שייצגו את המאזן החדש, ולבטל את TX A02 ואת TX B02 הישנים.

TX A2
Only Alice can broadcast

Outputs

- Local : 0.9 BTC
- Debit: 0.1 BTC

TX A3
Only Alice can broadcast

Outputs

- Local: 0.85 BTC
- Debit: 0.15 BTC

TX B2
Only Bob can broadcast

Outputs

- Remote: 0.9 BTC
- Credit: 0.1 BTC

TX B3
Only Bob can broadcast

Outputs

- Remote: 0.85 BTC
- Credit: 0.15 BTC

משמאל לימין, המאזנים של אליס ושל בוב בכל טרנזקציה

באופן דומה לאבטחה של Output מקומי ו-Output מרוחק, אנחנו צריכים להוסיף מנגנון ביטול (revocation) גם ל-Output חובה ול-Output זכות, והם ייראו כך:

Debit Output

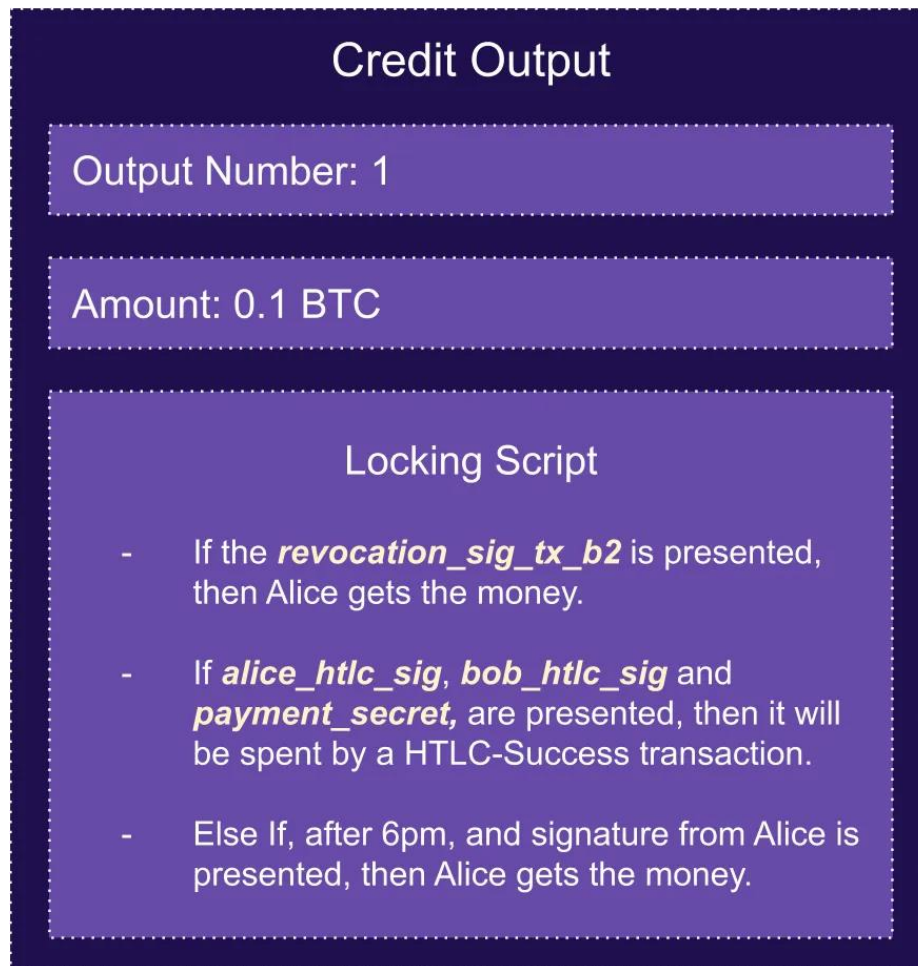
Output Number: 1

Amount: 0.1 BTC

Locking Script

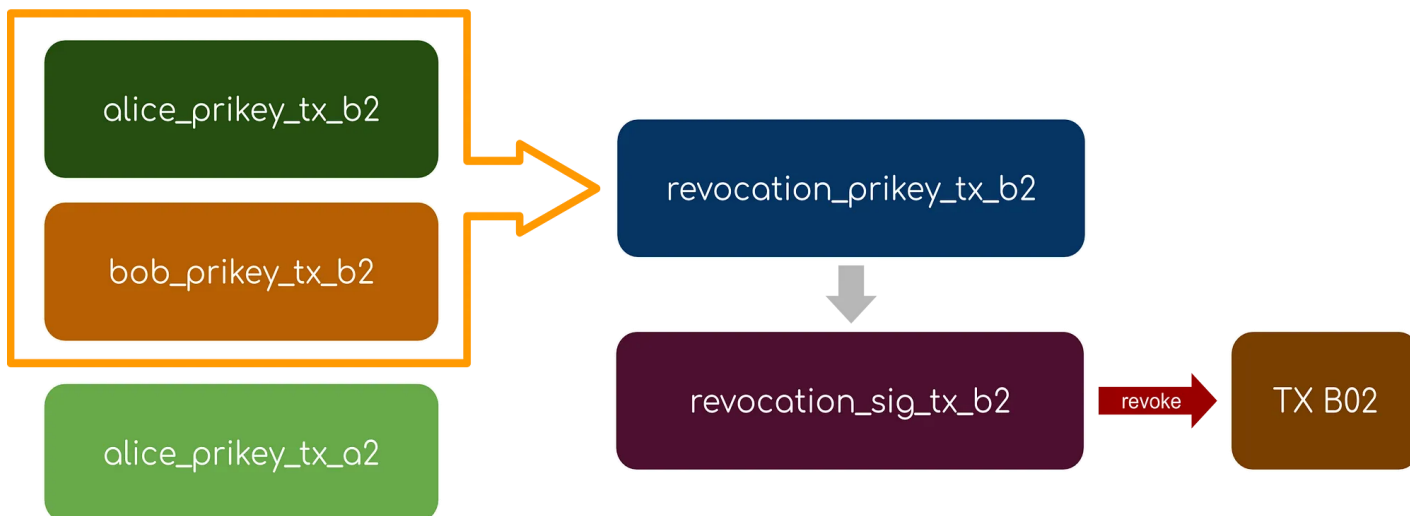
- If the **revocation_sig_tx_a2** is presented, then Bob gets the money.
- If **alice_htlc_sig** and **bob_htlc_sig** are presented, then it will be spent by a HTLC-Timeout transaction.
- Else if the signature from Bob and **payment_secret** are presented, then Bob gets the money.

Output החובה של אליס ב-TX A02

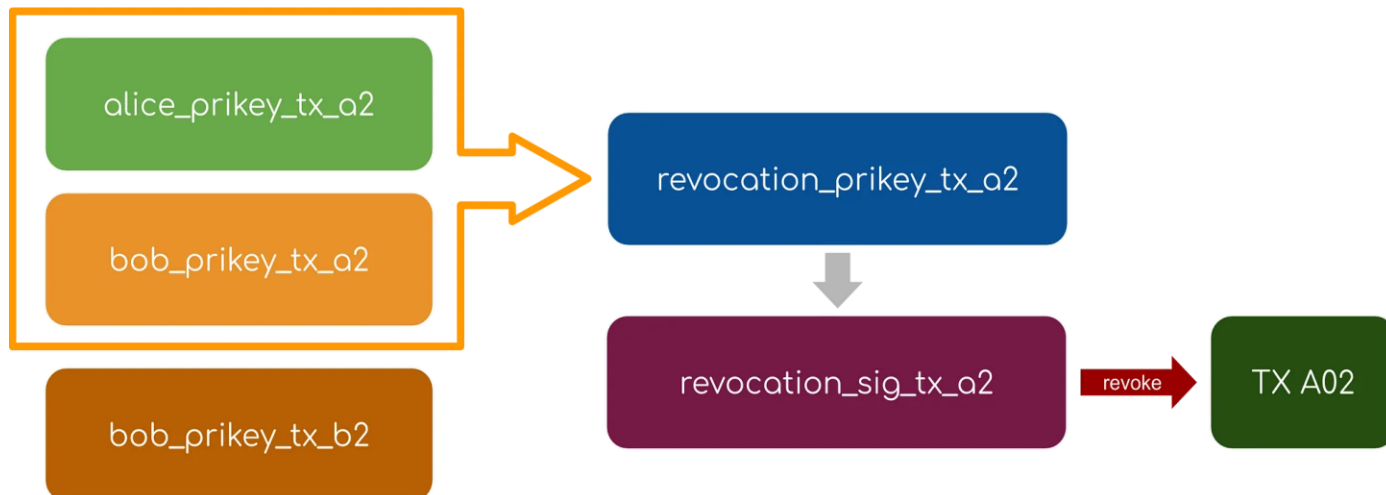


Output הזכות של בוב ב-TX B02

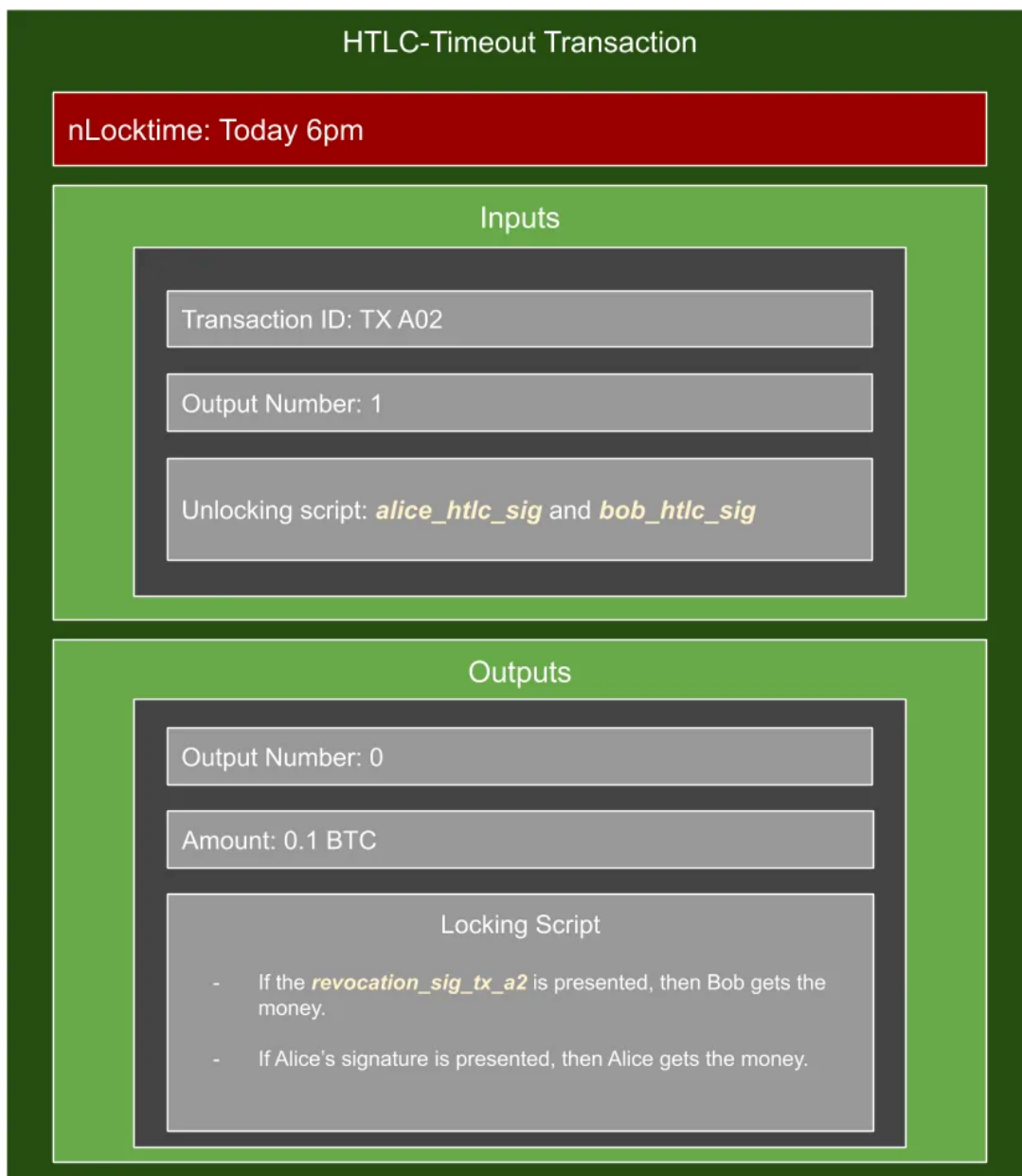
במהלך היצירה של TX A03 ושל TX B03, אליס ובוב ימסרו זה לזה את המפתחות הפרטיים הישנים - אליס תיתן לבוב את `alice_prikey_tx_a2` ובוב ייתן לאליס את `bob_prikey_tx_b2`. מבחינת TX A02, אם אליס תשדר אותה אז בוב (ורק בוב) יוכל "לבטל אותה" (לקחת את הכסף לעצמו), מכיוון שרק לו יש את `bob_prikey_tx_a2`. עבור TX B02, המנגנון הפוך - אם בוב משדר אותה, אז אליס (ורק אליס) תוכל "לבטל אותה" (לקחת את הכסף לעצמה), מכיוון שרק לה יש את `alice_prikey_tx_b2`.



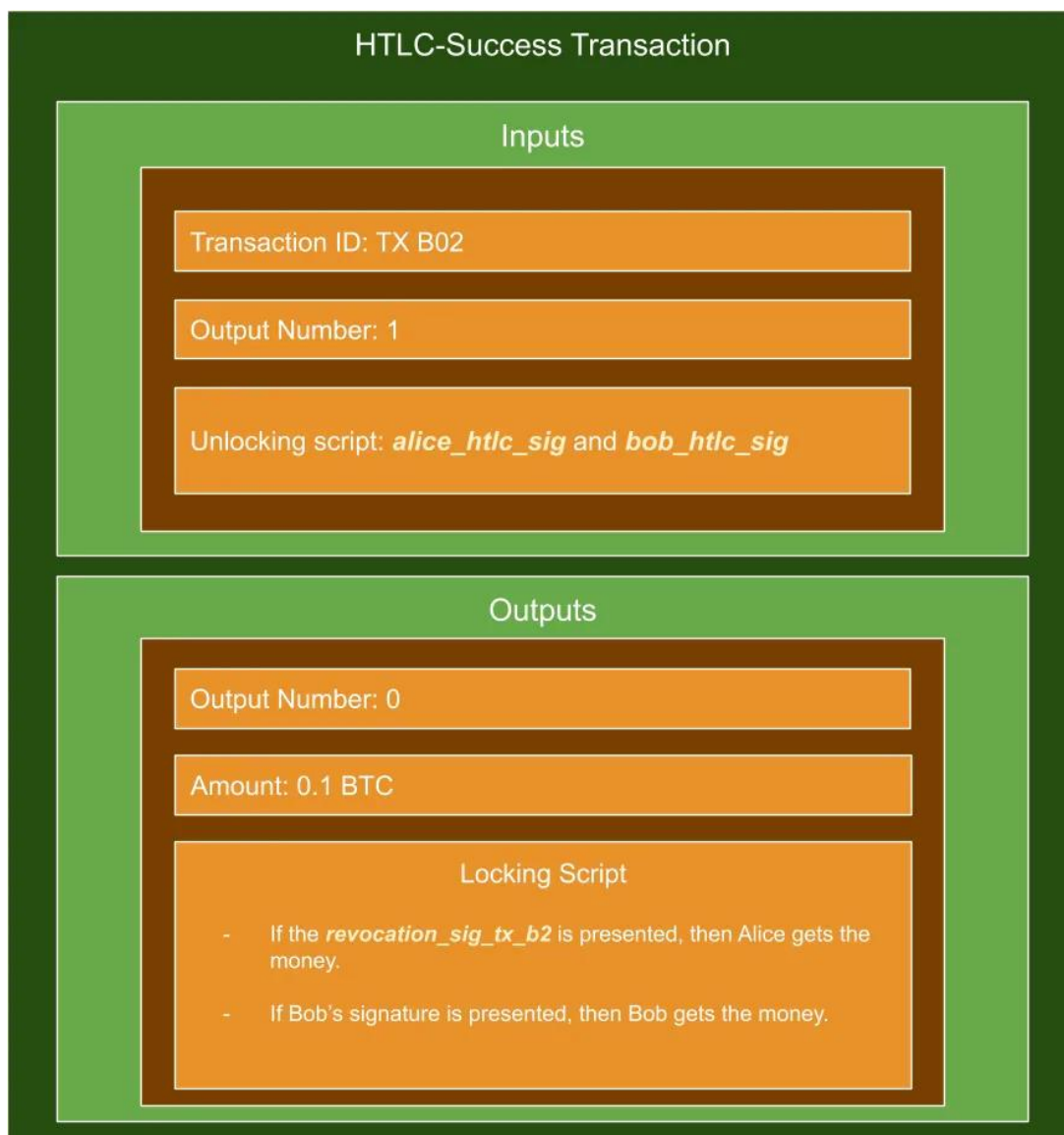
לאליס יש מראש את **alice_prikey_tx_b2**. אם היא קיבלה מבוב את **bob_prikey_tx_b2**, היא יכולה לחשב את **revocation_prikey_tx_b2**, שמייצר חתימה דיגיטלית **revocation_sig_tx_b2**, שיכולה לבטל את **TX B02**. אם בוב מרמה ומשדר את **TX B02**, כל הכסף ב־Output הזכות של הטרנזקציה יילקח מיידית ע"י אליס - בעונש לבוב.



לבוב יש מראש את **bob_prikey_tx_a2**. אם הוא קיבל מאליס את **alice_prikey_tx_a2**, הוא יכול לחשב את **revocation_prikey_tx_a2**, שמייצר חתימה דיגיטלית **revocation_sig_tx_a2**, שיכולה לבטל את **TX A02**. אם אליס מרמה ומשדרת את **TX A02**, כל הכסף ב־Output החובה של הטרנזקציה יילקח מיידית ע"י בוב - בעונש לאליס. מנגנון הביטול (revocation) נבנס גם בתוך טרנזקציות ה־HTLC-Timeout וה־HTLC-Success.



טרנזקציית ה-HTLC-Timeout של אליס. הערת מתרגם: בסקריפט הנעילה אליס לא מקבלת את הכסף מייד, אלא רק כעבור שבועיים, כדי לתת לבוב זמן להגיב.



טרנזקציית ה-HTLC-Success של בוב. הערת מתרגם, בסקריפט הנעילה בוב לא מקבל את הכסף מייד, אלא רק כעבור שבועיים, כדי לתת לאלים זמן להגיב.

למה מנגנון הביטול נכנס בשני מקומות, גם בטרנזקציות ההתחייבות וגם בטרנזקציות HTLC-Timeout/Success?

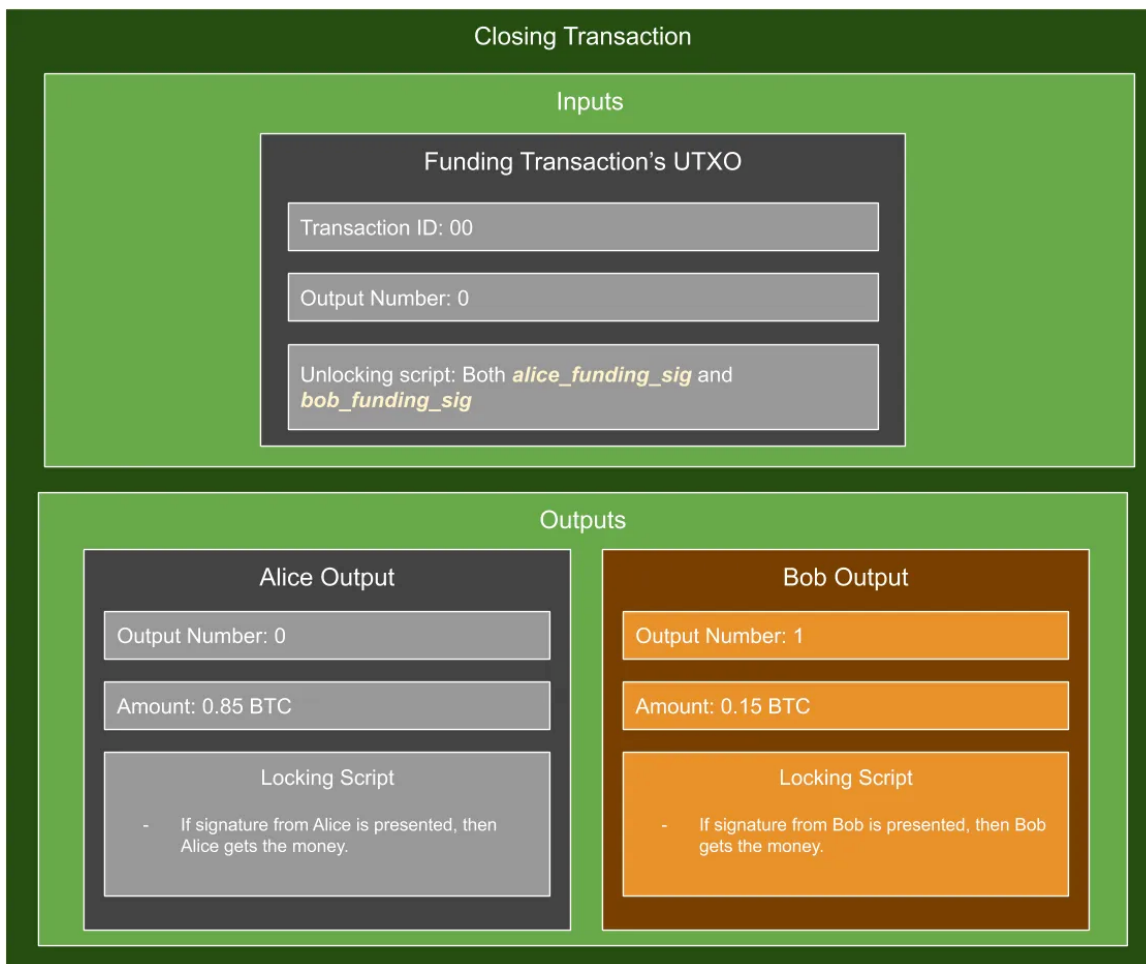
אם היה לנו מנגנון ביטול רק ב-Output החובה, אליס הייתה יכולה ב-18:00 לשדר גם את טרנזקציית TX Ao2 וגם את טרנזקציית ה-HTLC-Timeout. נכון שלבוב יש את היכולת לבטל את TX Ao2 אבל בזמן שייקח לו להגיב, רוב הכורזת ייראו שיש כבר טרנזקציה שמבזזת את אותו Output חובה - טרנזקציית ה-HTLC-Timeout. בוב יהיה תלוי בחסדיהם של הכורזת להעדיף את הטרנזקציה שלו על פני טרנזקציית ה-HTLC-Timeout.

מצד שני, אם היינו משאירות את הביטול רק בטרנזקציות ה-HTLC-Timeout/Success, ולא מכניסות אותו ל-Output החובה של אליס, היא הייתה יכולה לשדר את TX Ao2 בלי לשדר את טרנזקציית ה-HTLC-Timeout, ולגרום לכסף של בוב להינעל לתקופות ארוכות (ראו [דין](#)).

סגירת הערוץ - חזרה לרשת הביטקוין

ישנן [שלוש דרכים לסגור ערוץ](#). אליס ובוב יכולות:

1. הדרך הטובה: סגירה בשיתוף פעולה. אליס ובוב יכולות ליצור טרנזקציית סגירה, ולאסוף את הכסף שלהן מייד.
2. הדרך הרעה: סגירה חד צדדית באמצעות נעילת-זמן (ו-`payment_secret` אם צריך), שנחשבת רעה מכיוון שהכסף נשאר נעול לתקופה מסוימת. אליס יכולה לשדר את TX Ao3 ובוב יכול לשדר את TX Bo3 כדי לאסוף את הכסף שלהן, אבל הן יצטרכו לחכות שבועיים כדי להשתמש בו.
3. הדרך המכוערת: ביטול טרנזקציית רמאות של המשתתף השני. אם הצד השני מרמה ומשדר טרנזקציה ישנה, אפשר (וחובה) להשתמש ב-`revocation-key` כדי לאסוף את הכסף, בעונש לצד השני.



אליס ובוב יחתמו על טרנזקציית הסגירה ביחד, לאחר משא ומתן למזעור העמלות, וישדרו אותה.

המנגנון הסופי - העברת תשלומים

מאמץ גדול הושקע בתכנון של HTLC כדי שרשת הברק תוכל להעביר תשלומים באופן חלק. נניח שיש לנו עכשיו שני ערוצים: הראשון בין אליס לבוב שאליס תקצבה ב־0.1 ביטקוין, והשני בין בוב לצ'רלי שבו תקצב ב־0.2 ביטקוין. אליס רוצה לשלוח לצ'רלי 0.1 ביטקוין דרך בוב. הנה מה שהק צריכות לעשות:

בין בוב לצ'רלי:

1. צ'רלי מייצר `payment_charlie_secret`, ומפעיל עליו את פונקציית הגיבוב כדי לקבל `payment_charlie_hash`.
2. צ'רלי ובוב מייצרים טרנזקציות התחייבות דומות - לצ'רלי יש עכשיו Output זכות (HTLC נכנס) של 0.1 ביטקוין, ולבוב יש Output חובה (HTLC יוצא) של 0.1 ביטקוין.
3. צ'רלי חייב לחשוף את ה־`payment_charlie_secret` לבוב לפני השעה 17:00, אחרת בוב יקבל את כספו חזרה.
4. צ'רלי ובוב יוצרים גם טרנזקציות HTLC-Timeout ו-HTLC-Success דומות.

Commitment Transaction: TX Bob-Charlie

Inputs

Funding Transaction's UTXO

Transaction ID: xxx

Output Number: 0

Unlocking script: Both *bob_funding_sig* and *charlie_funding_sig*

Outputs

Local Output

Output Number: 0

Amount: 1.9 BTC

Locking Script

- If two weeks have passed and signature from Bob is presented, then Bob gets the money.
- Else if the *revocation_sig_tx_bc* is presented, then Charlie gets the money.

Debit Output

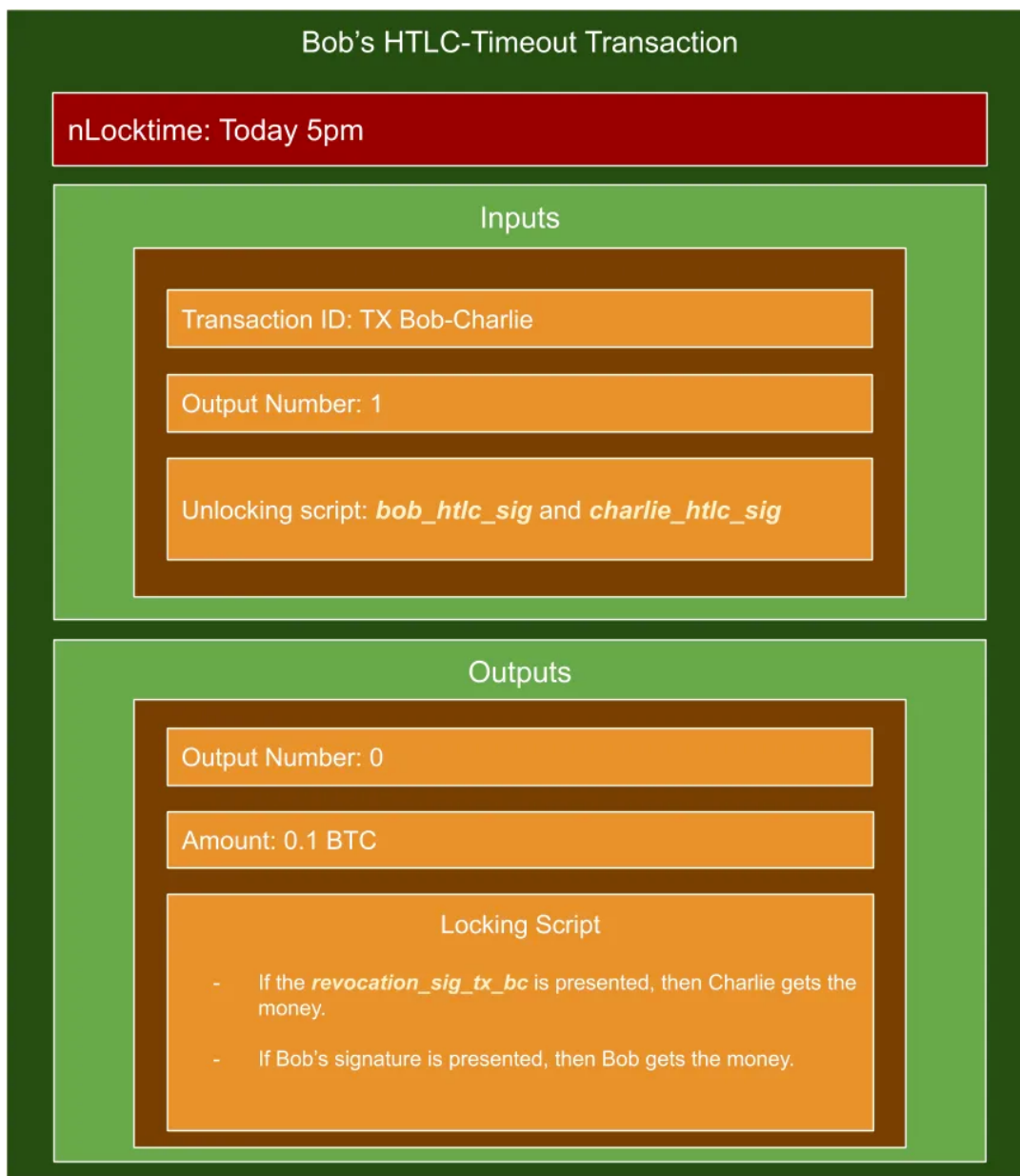
Output Number: 1

Amount: 0.1 BTC

Locking Script

- If the *revocation_sig_tx_bc* is presented, then Bob gets the money.
- If *bob_htlc_sig* and *charlie_htlc_sig* are presented, then it will be spent by a HTLC-Timeout transaction.
- Else if the signature from Charlie and *payment_charlie_secret* are presented, then Charlie gets the money.

טרנזקציית ההתחייבות החדשה של בוב, שבה הוא שולח 0.1 ביטקוין לצ'רלי.



טרנזקציית ה-HTLC-Timeout של בוב, שאותה הוא יכול לשדר החל מהשעה 17:00. הערת מתרגם: גם כאן שבחו לציין בסקריפט הנעילה שבוב יכול לקבל את כספו רק אחרי שבועיים - כדי לתת זמן לצ'רלי להגיב אם הוא יכול לייצר את *revocation_sig_tx_bc*.

Commitment Transaction: TX Charlie-Bob

Inputs

Funding Transaction's UTXO

Transaction ID: xxx

Output Number: 0

Unlocking script: Both *bob_funding_sig* and *charlie_funding_sig*

Outputs

Remote Output

Output Number: 0

Amount: 1.9 BTC

Locking Script

- If signature from Bob is presented, then Bob gets the money.

Credit Output

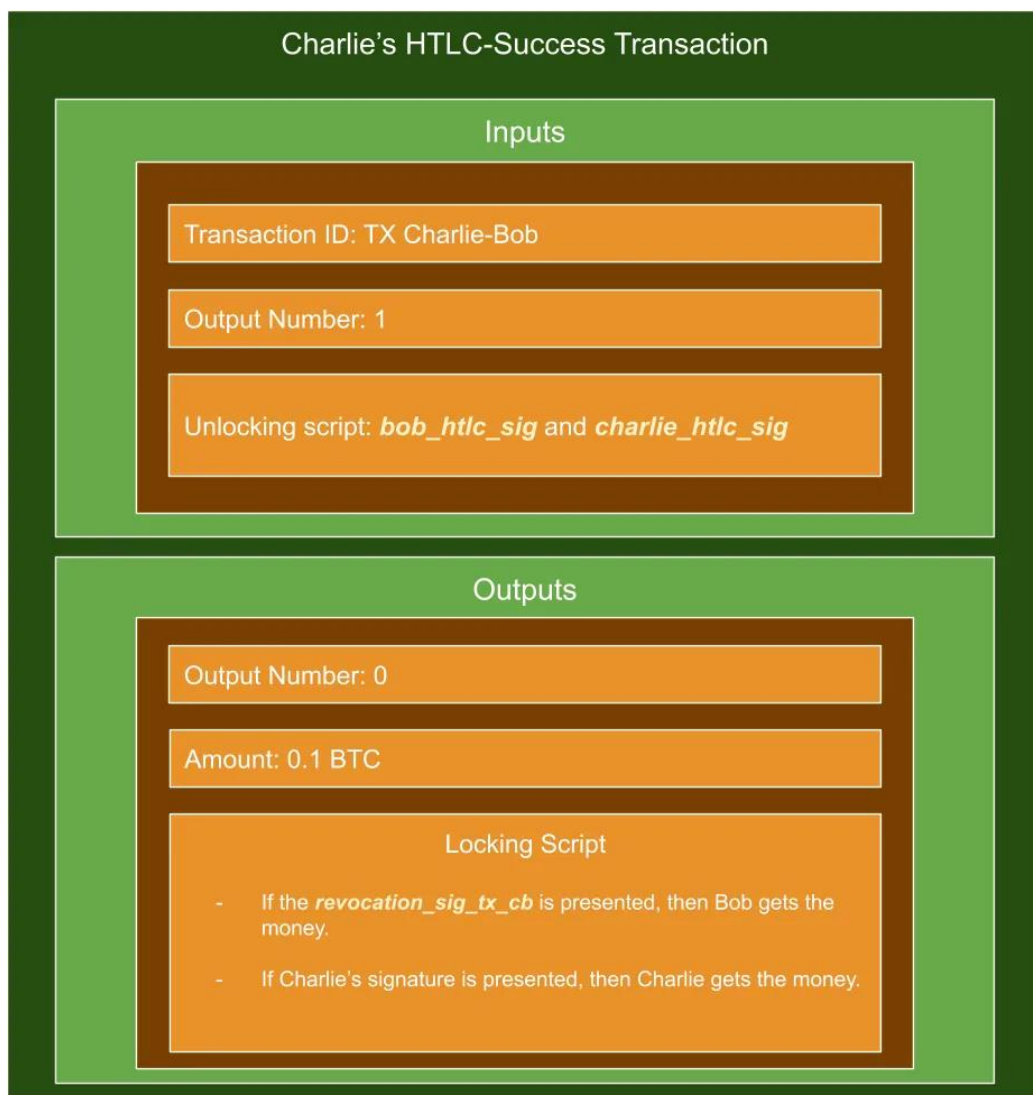
Output Number: 1

Amount: 0.1 BTC

Locking Script

- If the *revocation_sig_tx_cb* is presented, then Bob gets the money.
- If *bob_htlc_sig*, *charlie_htlc_sig* and *payment_charlie_secret* are presented, then it will be spent by a HTLC-Success transaction.
- Else If, after 5pm, and signature from Bob is presented, then Bob gets the money.

טרנזקציית ההתחייבות של צ'רלי, שבה הוא מקבל 0.1 ביטקוין מבוב.



טרנזקציית ה-HTLC-Success של צ'רלי. הערת מתרגם: גם כאן שנחו לציין בסקריפט הנעילה שצ'רלי יכול לקבל את כספו רק אחרי שבועיים - כדי לתת זמן לבוב להגיב אם הוא יכול לייצר את *revocation_sig_tx_cb*. בין אליס ובוב, הק יחזרו על התהליך, כאשר בוב הוא זה שתפקידו "לאסוף כסף". טרנזקציות ההתחייבות שלהם יהיו:

- לבוב יש Output זכות (ה-HTLC הנכנס) של 0.1 ביטקוין.
 - לאליס יש Output חובה (ה-HTLC היוצא) של 0.1 ביטקוין.
 - בוב צריך לחשוף את `payment_charlie_secret` לפני 18:00 (שעה אחרי ה-deadline בעסקה שלו מול צ'רלי). אחרת אליס מקבלת את הכסף חזרה.
- בוב ואליס גם ייצרו טרנזקציות HTLC-Timeout ו-HTLC-Success דומות. כדי להשלים את התשלום:
1. צ'רלי יחשוף את `payment_charlie_secret` לבוב לפני 17:00, כדי לקבל את הכסף שלו מבוב.

2. לבוב תהיה לפחות שעה שלמה לחשוף את `payment_charlie_secret` כדי לקבל את בספו מאליס.

במציאות, בוב, שרק מעביר הלאה את התשלום, יכול לדרוש מאליס עמלה קטנה עבור השירות שהוא נותן (כלומר, אليس תשלם לבוב קצת יותר מאשר מה שצ'רלי דרש מבוב). הטכנולוגיה לכך כבר בשימוש היום, אבל יהיה מרתק לראות בעתיד אילו השפעות פוליטיות וכלכליות יבואו לידי ביטוי במנגנוני העמלות הללו!

הערות מתרגם:

א. בוב ישלח לצ'רלי את החתימות הדיגיטליות הרלוונטיות רק אחרי שקיבל את החתימות שהוא צריך מאליס. בוב הרי לא רוצה להבטיח לצ'רלי 0.1 ביטקוין תמורת אותו `payment_charlie_secret`, לפני שהוא יודע שהוא יכול לקבל מאליס את אותו הסכום תמורת אותו `payment_charlie_secret`.

ב. ראוי לציין שברשת הברק יש מנגנוני פרטיות נוספים שלא מצוינים פה: בוב לא באמת יודע אם התשלום התבצע מאליס לצ'רלי, או ממקור שנמצא לפני אليس אל יעד שנמצא אחרי צ'רלי. בוב רק יודע שאליס וצ'רלי עושים עסקה להעברת `payment-secret`, בלי לדעת אם מדובר בשרשרת ארוכה של עסקאות כאלה שהוא רק חלק קטן ממנה.

הבהרה לסיום – עבור מפתח/חוקרית:

בשביל הפשטות, קיצרתי במכוון את סקריפט הנעילה כדי להתרכז בתמונה הגדולה. אני חושב שיהיה נחמד להבהיר מהו הסקריפט שמשתמשת בו בפועל. כשאנו מדברות על הפעולה "לוודא שהחתימה של אليس נמצאת" בסקריפט נעילה, מה שנכנס לסקריפט זה:

`<Alice's public key> OP_CHECKSIG`

כאשר אليس מבזבזת את ה-Output, היא שמה ב-Input של הטרנזקציה הבאה את `Alice's public key`. אותו דבר קורה עבור חתימת `revocation` (ביטול) ועבור `payment-secret`. לפרטים נוספים, ראו: הערות ומקורות ידע לפרק ב'.

הבהרה נוספת היא האלגוריתם שבו משתמשים כדי ליצור את "זוגות מפתחות הביטול" (`revocation key pairs`). קודם לכן, תיארנו את זה כחיבור פשוט של המפתחות, אבל המתמטיקה בפועל היא כדלקמן:

$$\begin{aligned} \text{revocation_public_key} &= \text{bob_public_key} * \text{SHA256}(\text{bob_public_key} || \text{alice_public_key}) \\ &+ \text{alice_public_key} * \text{SHA256}(\text{alice_public_key} || \text{bob_public_key}) \\ \text{revocation_private_key} &= \text{bob_secret_key} * \text{SHA256}(\text{bob_public_key} || \\ &\text{alice_public_key}) + \text{alice_secret_key} * \text{SHA256}(\text{alice_public_key} || \text{bob_public_key}) \end{aligned}$$

אם נרצה להתאים את שמות המפתחות ל-RFC, נניח שבוט הוא הצומת המקומי (local node) ואליס היא הצומת המרוחק (remote node), ונחליף את השמות כך:

- `bob_public_key` הוא `revocation_basepoint`.
- `bob_secret_key` הוא `revocation_basepoint_secret`.
- `alice_public_key` הוא `per_commitment_point`.
- `alice_secret_key` הוא `per_commitment_secret`.

הנוסחאות שנקבל יהיו זהות.

	This is the unlocking script, kept in secret by each party.	This is the locking script used in commitment transactions and HTLC transactions.
Check signature:	<Alice's signature>	<Alice's public key> OP_CHECKSIG
Check revocation signature:	<revocation_sig> <revocation_pubkey>	OP_DUP OP_HASH160 <revocation_pubkey> OP_EQUALVERIFY OP_CHECKSIG
Check multisig:	0 <alice_funding_sig> <bob_funding_sig>	2 <alice_funding_public_key> <bob_funding_public_key> 2 OP_CHECKMULTISIG
Check payment secret:	<payment_secret>	OP_HASH160 <payment_hash> OP_EQUALVERIFY

הטקסטים שצבועים בירוק בתאים האדומים (סקריפט נעילה) משותפים בין כולם.

הערות ומקורות ידע לפרק ב'

1. [The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments](#), Joseph Poon, Thaddeus Dryja
2. [Reaching the ground with lightning](#), Rusty Russell
3. [Elliptic Curve Point Multiplication](#)
4. [The Lightning Network Specifications](#) is a great place to find all the details, though it's difficult to read. Regarding how transactions work, [BOLT #2: Peer Protocol for Channel Management](#), [BOLT #3: Bitcoin Transaction and Script Formats](#) and [BOLT #5: Recommendations for On-chain Transaction Handling](#) explain it in technical detail.
5. Rene Pickhardt, who works with Andreas M. Antonopoulos to write the book, [Mastering the Lightning Network](#). He has helped to explain the [general process](#)

[of the transaction](#), [how the keys exchanged](#) and created [this presentation](#), which I found very helpful.

6. Hongchao wrote [Payment Channels in Lightning Network](#) which gave an overview of how the lightning network works, it's easy to understand for readers with knowledge on Bitcoin script.
7. Rusty Russell (the main contributor to the lightning-RFC) has summarized the rationality behind the two-stage HTLC outputs design in his blog, [The #Bitcoin #Lightning Spec Part 3/8: Peer Protocol for Channel Management](#).
8. About the timelock, James Prestwich's [Bitcoin's Time Locks](#) has provided great insight, in addition to the [Bitcoin Developer Guide](#) and Andreas M. Antonopoulos's Mastering Bitcoin, [Advanced Transactions and Scripting](#).
9. I've set up a testing environment using [bitcoind](#) and [lnd](#) to help me understand the whole process. Here's [the gist of the trimmed log](#) generated from lnd, which recorded how Alice sent Bob money.
10. A clarification on the locking scripts. The red boxes represent the actual format of the locking script. To unlock it, you need to supply the unlocking script in the corresponding green box.