Authenticated Key Exchange over Bitcoin

Patrick McCorry(✉), Siamak F. Shahandashti, Dylan Clarke, and Feng Hao

School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
{patrick.mccorry,siamak.shahandashti,dylan.clarke,
feng.hao}@ncl.ac.uk

Abstract. Bitcoin is designed to protect user anonymity (or pseudonymity) in a financial transaction, and has been increasingly adopted by major e-commerce websites such as Dell, PayPal and Expedia. While the anonymity of Bitcoin transactions has been extensively studied, little attention has been paid to the security of post-transaction correspondence. In a commercial application, the merchant and the user often need to engage in follow-up correspondence after a Bitcoin transaction is completed, e.g., to acknowledge the receipt of payment, to confirm the billing address, to arrange the product delivery, to discuss refund and so on. Currently, such follow-up correspondence is typically done in plaintext via email with no guarantee on confidentiality. Obviously, leakage of sensitive data from the correspondence (e.g., billing address) can trivially compromise the anonymity of Bitcoin users. In this paper, we initiate the first study on how to realise end-to-end secure communication between Bitcoin users in a post-transaction scenario without requiring any trusted third party or additional authentication credentials. This is an important new area that has not been covered by any IEEE or ISO/IEC security standard, as none of the existing PKI-based or password-based AKE schemes are suitable for the purpose. Instead, our idea is to leverage the Bitcoin's append-only ledger as an additional layer of authentication between previously confirmed transactions. This naturally leads to a new category of AKE protocols that bootstrap trust entirely from the block chain. We call this new category "Bitcoin-based AKE" and present two concrete protocols: one is non-interactive with no forward secrecy, while the other is interactive with additional guarantee of forward secrecy. Finally, we present proof-of-concept prototypes for both protocols with experimental results to demonstrate their practical feasibility.

Keywords: Authenticated key exchange · Bitcoin · Diffie-Hellman · YAK

1 Introduction

Bitcoin [22] is an online currency whose value is not endorsed by any central reserve, but is based on the perception of its users [15]. In recent years it has surged in value, reaching a peak of $1147 per bitcoin in December 2013.

L. Chen and S. Matsuo (Eds.): SSR 2015, LNCS 9497, pp. 3–20, 2015.
DOI: 10.1007/978-3-319-27152-1_1

The currency is supported by a decentralised network of users whose collective computational power provides a guarantee of integrity for an append-only ledger. Any attempt to change the ledger's history (a history-revision attack [4]) would require an adversary with at least, in theory 51 % of the networks computational resources to be successful[1]. Several central banks have evaluated the value of digital currencies and their potential impact on society [15,25,29].

Bitcoin is increasingly being accepted by many e-commerce websites as a form of payment. For example, Dell, one of the largest computer retailers in the world, now allows customers to use Bitcoin to pay for online purchases on the Dell website [9]. Recently, PayPal [5] and Expedia [24] have also endorsed support for using Bitcoin. Similarly, many community-driven organisations allow anonymous donations using Bitcoin. Examples include the TOR project [27], Mozilla Foundation [21] and the Calyx Institute [18],

While Bitcoin is designed to support anonymity (or pseudonymity) in a transaction, little attention has been paid to the anonymity in the post-payment scenario. As with any on-line payment system, the payer and the payee may need to engage in follow-up correspondence after the payment has been made, e.g., to acknowledge the receipt, to confirm billing information, to amend discrepancies in the order if there are any and to agree on the product delivery or pick-up. Such correspondence can involve privacy-sensitive information, which, if leaked to a third party, may trivially reveal the identity of the user involved in the earlier transaction (e.g., information about product delivery may contain the home address).

Currently, the primary mechanism to support follow-up correspondence after a Bitcoin transaction is through email. The Dell website requires shoppers to provide their email address when making a Bitcoin payment to facilitate post-payment correspondence. The Calyx Institute, a non-profit research organization dedicated to providing "privacy by design for everyone", also recommends using e-mails for follow-up correspondence after a donation is made in Bitcoin. On its website, the instruction is given as the following [18]:

> *"Note that if you make a donation by Bitcoin, we have no way to connect the donation with your email address. If you would like us to confirm receipt of the donation (and send a thank you email!), you'll need to send an email with the details of the transaction. Otherwise, you have our thanks for your support in advance".*

However, emails are merely a communication medium and have no built-in guarantees of security. First of all, there is no guarantee that the sender of the email must be the same person who made the Bitcoin payment. The details of the transaction cannot serve as a means of authentication, since they are publicly available on the Bitcoin network. Furthermore, today's emails are usually not encrypted. The content of an email can be easily read by third parties (e.g., ISPs) during the transit over the Internet. The leakage of privacy-sensitive information

[1] An adversary may not require 51 % of computational power in reality [3,4,10].

in email can seriously threaten the anonymity of the user who has made an "anonymous" payment in Bitcoin previously.

So far the importance of protecting post-payment communication has been largely neglected in both the Bitcoin and the security research communities. To the best of our knowledge, no solution is available to address this practical problem in the real world. This is a gap in the field, which we aim to bridge in our work.

One trivial solution is to apply existing Authenticated Key Exchange (AKE) protocols to establish a secure end-to-end (E2E) communication channel between Bitcoin users. Two general approaches for realising secure E2E communication in cryptography include using 1) PKI-based AKE (e.g., HMQV), and 2) Password-based AKE (e.g., EKE and SPEKE). The former approach would require Bitcoin users to be part of a global PKI system, with each user holding a public key certificate. This is not realistic in current Bitcoin applications. The second approach requires Bitcoin users to have a pre-shared secret password. However, securely distributing pairwise shared passwords over the internet is not an easy task. Furthermore, passwords are a weak form of authentication and they may be easily guessed or stolen (e.g. by shoulder-surfing). A solution that can provide a stronger form of authentication without involving any passwords will be desirable.

Following the decentralised and anonymity-driven nature of the Bitcoin network [17], we propose new AKE protocols to support secure post-payment communication between Bitcoin users, without requiring any PKI or pre-shared passwords. Our solutions leverage the transaction-specific secrets in the confirmed Bitcoin payments published on the public blockchain to bootstrap trust in establishing an end-to-end secure communication channel. Given each party's transaction history and our AKE protocols, both parties are guaranteed to be speaking to the other party who was involved in the transactions, without revealing their real identities.

Contributions. Our contributions in this paper are summarised below.

- We propose two authenticated key exchange protocols – one interactive and the other non-interactive – using transaction-specific secrets and without the support of a trusted third party to establish end-to-end secure communication. These are new types of AKE protocols, since they bootstrap trust from Bitcoin's public ledger instead of a PKI or shared passwords.
- We provide proof-of-concept implementations for both protocols in the Bitcoin Core client with performance measurements. Our experiments suggest that these protocols are feasible for practical use in real-world Bitcoin applications.

Organization. The rest of the paper is organised as follows. Section 2 explains the background of Bitcoin and the ECDSA signature that is used for authenticating Bitcoin transactions. Section 3 proposes two protocols to allow post-payment secure communication between users based on their transaction history. One protocol is non-interactive with no forward secrecy, while the other is interactive with the additional guarantee of forward secrecy. Security proofs for both

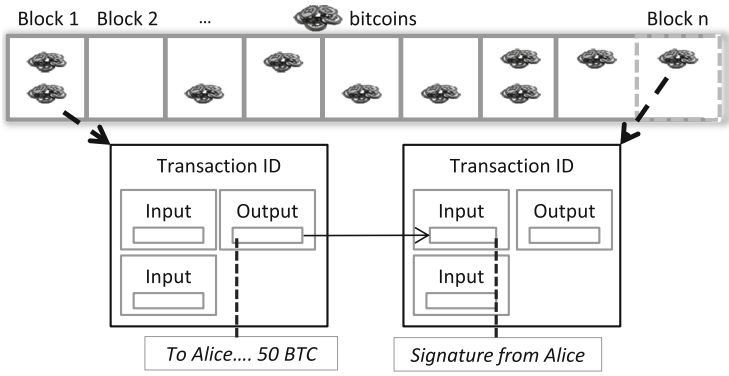

Fig. 1. Transactions stored on the Blockchain based on [19]

protocols are provided in Sect. 4. Section 5 describes the proof-of-concept implementations for both protocols and reports the performance measurements. Finally, Sect. 6 concludes the paper.

2 Background

In this section, we will provide brief background information about the Bitcoin protocol, the transaction signature and the underlying Elliptic Curve Digital Signature Algorithm (ECDSA). This information will be needed for understanding the two protocols presented in this paper.

2.1 Bitcoin

Bitcoin is a digital currency that allows a network of nodes to access a public ledger and to agree upon append-only changes periodically. We will outline the three main mechanisms in the Bitcoin protocol which include Bitcoin addresses, transactions and the Blockchain. Together, they allow users to identify each other pseudonymously, transfer bitcoins and record the transaction in the public ledger.

Each user is responsible for generating their Bitcoin address, which is simply the hash of an ECDSA public key. The corresponding private key is required to spend bitcoins. This approach for user identification is considered appropriate as the probability that two users generate the same public key is negligible due to the high number of possible ECDSA public keys.

A common belief in the community is that Bitcoin offers *pseudonymity* that can help disguise their real-world identity due to the random nature of ECDSA public keys. This belief is bolstered as users are recommended to create a new Bitcoin address per transaction to increase the difficulty of tracking their transactions. However, it should be noted that Bitcoin was not designed with *anonymity*

Algorithm 1. ECDSA Signature Generation algorithm [11]

Input: Domain parameters $D = (q, P, n, \text{Curve})$, private key d, message m.
Output: Signature (r, s).

1: Select $k \in_R [1, n-1]$.
2: Compute $kP = (x_1, y_1)$ where $x_1 \in_R [0, q-1]$
3: Compute $r = x_1 \bmod n$. If $r = 0$, then go to Step 1.
4: Compute $e = H(m)$.
5: Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$, then go to Step 1.
6: Return (r, s).

in mind [23] and studies have shown with limited success that it is possible to link Bitcoin addresses to real-world identities [3,23,26].

Transactions are created by users to send bitcoins. All transactions are sent to the network and its correctness is verified by other peers before it is accepted into the public ledger. Each transaction has a list of 'inputs' and 'outputs'. The output states the new owner's bitcoin address and the quantity to be transferred. The input will contain a signature to authorise the payment and a reference to a previous transaction whereby the user received the bitcoins. Figure 1 highlights how transactions are linked, which allows peers to perform the verification, by comparing the received transaction with their local copy of the ledger.

A special 'miner' will collect the most recent set of transactions from the network to form a 'block'. This block is appended to the longest chain of blocks (Blockchain) approximately every ten minutes by solving a computationally difficult problem (proof of work) in return for a subsidy of bitcoins. This append-only ledger has become a relatively secure time stamp server [7], since reversing transactions that are committed on the Blockchain is considered infeasible. Figure 1 demonstrates how transactions are stored aperiodically on the Blockchain.

2.2 Transaction Signature

Figure 1 presented earlier demonstrates that the signature is stored in the input of a transaction. This signature must be from the Bitcoin address mentioned in the previous transaction's output. Briefly, it is important to highlight that the user will create the transaction, specify the inputs and outputs, hash this transaction and then sign it using their private key. This prevents an adversary from modifying the contents of a transaction or claiming ownership of the bitcoins before it is accepted into the Blockchain.

Bitcoin incorporates the OpenSSL suite to execute the ECDSA algorithm. The NIST-P256 curve is used and all domain parameters over the finite field including group order n, generator P and modulus q can be found in [6]. An outline of the signature generation algorithm is presented in Algorithm 1 to highlight the usage of k as this will be required for the authenticated key exchange protocols. The verification algorithm follows what is defined in [13]. The notations and symbols used in our paper are summarised in Table 1.

3 Key Exchange Protocols

Key exchange protocols allow two or more participants to derive a shared cryptographic key, often used for authenticated encryption. In this section we will present two authenticated key exchange protocols: Diffie-Hellman-over-Bitcoin and YAK-over-Bitcoin. These protocols will take advantage of a random nonce k from an ECDSA signature. Our aim is to achieve transaction-level authentication by taking advantage of a secret that only exists due to the creation of a transaction that is stored on the Blockchain.

Both of these protocols will use k as a transaction-specific private key and $Q = kP$ as a transaction-specific public key. Diffie-Hellman-over-Bitcoin will be a non-interactive protocol without forward secrecy and YAK-over-Bitcoin will be an interactive protocol with forward secrecy. All domain parameters D for both protocols are the same as the ECDSA algorithm.

Table 1. Summary of notations and symbols

$ZKP\{w\}$	Zero knowledge proof of knowledge of w
(V, z)	Schnorr zero knowledge proof values
$KDF(.)$	Key derivation function
$Uncompress(x, sign)$	Uncompresses public key using x co-ordinate and $sign \in \{+, -\}$
(x, y)	Represents a point on the elliptic curve
P	Generator for the elliptic curve
(r, s)	Signature pair that is stored in a transaction
A, B	Alice and Bob's bitcoin addresses: $H(dP)$
$d_{\mathrm{A}}, d_{\mathrm{B}}$	Alice and Bob's private key for their Bitcoin address
$k_{\mathrm{A}}, k_{\mathrm{B}}$	Alice and Bob's transaction-specific private key
$\widehat{k}_{\mathrm{A}}, \widehat{k}_{\mathrm{B}}$	Alice and Bob's estimated transaction-specific private key
$Q_{\mathrm{A}}, Q_{\mathrm{B}}$	Alice and Bob's transaction-specific public key
$\widehat{Q}_{\mathrm{A}}, \widehat{Q}_{\mathrm{B}}$	Alice and Bob's estimated transaction-specific public key
$w_{\mathrm{A}}, w_{\mathrm{B}}$	Alice and Bob's ephemeral private keys used for YAK
κ_{AB}	Shared key for Alice and Bob

3.1 Setting the Stage

We will have two actors, Alice and Bob. A single transaction T_A is used by Alice to send her payment (anonymously or not) to Bob. For our protocols, we will assume that Bob has created a second transaction T_b using his ECDSA private key, so the Blockchain contains both Alice and Bob's ECDSA signature. This is a realistic assumption as Bob naturally needs to spend the money or re-organise his bitcoins to protect against theft. In one possible implementation, upon receiving Alice's payment, Bob can send back to Alice a tiny portion of the received amount as acknowledgement, so his ECDSA signature is published on the blockchain (the signature serves to prove that Bob knows the ECDSA private key). This is just one way to ensure that the Blockchain contains both actors' signatures, and there may be many other methods to achieve the same.

Blockchain contains (r_A, s_A) *and* (r_B, s_B) *from* T_A *and* T_B	
Alice (A, d_A)	**Bob** (B, d_B)
1. $k_A = (H(T_A) + d_A r_A) s_A^{-1}$	$k_B = (H(T_B) + d_B r_B) s_B^{-1}$
2. $\widehat{Q}_B = Uncompress(r_B, +)$	$\widehat{Q}_A = Uncompress(r_A, +)$
3. $k_A \widehat{Q}_B = (x_{AB}, \pm y_{AB})$	$k_B \widehat{Q}_A = (x_{AB}, \pm y_{AB})$
$\kappa = \mathrm{KDF}(x_{AB})$	$\kappa = \mathrm{KDF}(x_{AB})$

Fig. 2. The Diffie-Hellman-over-Bitcoin Protocol

The owner of a transaction will be required to derive the transaction-specific private key (random nonce) k from their signature before taking part in the key exchange protocols. For both protocols, we assume the transactions T_A, T_B between Alice and Bob have been sent to the network and accepted to the Blockchain with a depth of at least six blocks, which is considered the standard depth to rule-out the possibility of a double-spend attack [14].

In both protocols, each user will need to extract their partner's signature (r, s) and attempt to derive their partner's transaction-specific public key $Q = (x, y)$. Algorithm 1 demonstrates that the r value from the signature is equal to the x co-ordinate modulo n (note that there is a subtle difference in the data range, since $r \in Z_n$ and $x \in Z_q$, but this has an almost negligible effect on the working of the protocols as we will explain in detail in Sect. 5.2). However, the y co-ordinate of Q is not stored in the transaction, and it can be either of the two values (above/below the x axis).

We define the uncompression function as $Uncompress(x, sign)$ by using the x co-ordinate from their partner's signature and the y co-ordinate's $sign \in \{+, -\}$. Using point uncompression and assuming one of the two possible signs for the y co-ordinate, Alice or Bob will be able to derive a value $\widehat{Q}$ which we call the estimated transaction-specific public key for their partner. This $\widehat{Q}$ could be either $Q = (x, y)$ or its additive inverse $-Q = (x, -y)$. This $\widehat{Q}$ will correspond to the estimated transaction-specific private key $\widehat{k}$, which could be either k or $-k$.

3.2 Authentication

Our definition of authentication will refer to data origin authentication and we will use the Blockchain as a trusted platform for storing digital signatures. Knowledge of the private key d for a bitcoin address or the random nonce k in a signature will prove the identities of pseudonymous parties. We will define two concepts for authentication using Bitcoin:

1. **Bitcoin address authentication**. Knowledge of the discrete log d for a Bitcoin address.
2. **Transaction authentication**. Knowledge of the discrete log k from a single digital signature in a transaction.

Bitcoin address authentication is well-known in the community and has been used for other protocols. However, transaction authentication is a special case

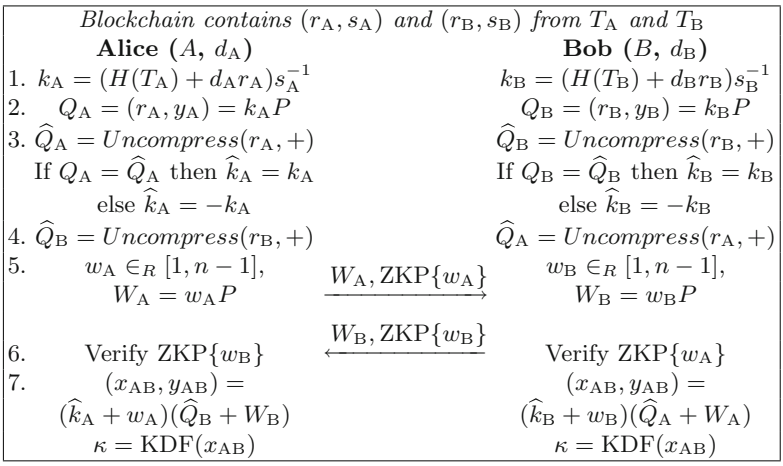

Fig. 3. YAK-over-Bitcoin Protocol

that our protocols will exploit. Although k and d are equivalent in proving ownership of a Bitcoin address or transaction, k is randomly generated for every ECDSA signature and is unique for each new transaction.

We will show that Alice and Bob can authenticate each other based on the knowledge of the k. This relies on participants trusting the integrity of the Blockchain as the cornerstone for authentication. For an adversary to mount a man-in-the-middle attack in this scene, he would need to perform a history-revision attack to modify the ECDSA signatures stored in the Blockchain.

3.3 Diffie-Hellman-over-Bitcoin Protocol

Based on the concept of transaction authentication, the first protocol that we present is 'Diffie-Hellman-over-Bitcoin'. The protocol is non-interactive; the shared secret is generated using the signatures from two transactions and no additional information from the participants is required. However, forward secrecy is not provided, as we will illustrate in the security analysis.

Figure 2 presents an outline of the protocol. Initially, each user will derive the random nonce k from their own signatures and fetch their partner's transaction from the Blockchain. Each user will gain an estimation of their partner's public key $\widehat{Q}$ before using their own transaction-specific private key k to derive the shared secret $(x_{AB}, \pm y_{AB})$. Regardless of whether $\widehat{Q}_A = \pm Q_A$ (or $\widehat{Q}_B = \pm Q_B$), the x co-ordinate of $k_B \widehat{Q}_A$ will be the same as that of $k_A \widehat{Q}_B$. Following the Elliptic Curve Diffie Hellman (ECDH) [20] approach, the x_{AB} co-ordinate will be used to derive the key $KDF(x_{AB}) = \kappa$.

Algorithm 2. Schnorr Zero Knowledge Proof Generation Algorithm

Input: Domain parameters $D = (q, P, n, \text{Curve})$, signer identity ID, secret value w and public value W.
Output: (V, z)

1: Select $v \in_R [1, n-1]$, Compute $V = vP$
2: Compute $h = H(D, W, V, ID)$
3: Compute $z = v - wh \mod n$
4: Return (V, z)

3.4 YAK-over-Bitcoin Protocol

The second protocol we present is 'YAK-over-Bitcoin'. This is based on adapting a PKI-based YAK key exchange protocol [12] to the Bitcoin application by removing the dependence on a PKI and instead relying on the integrity of the Blockchain. We chose YAK instead of others (e.g., station-to-station, MQV, HMQV, etc.), as YAK is the only PKI-based AKE protocol that requires each sender to demonstrate the proof of knowledge of both the static and ephemeral private keys. This requirement is important for the security proofs of our system as we will detail in Sect. 4. As well, we will show in the security analysis that the protocol allows the participants to have full forward secrecy.

An outline of our protocol is presented in Fig. 3. Initially, each user will follow the same steps as seen in the previous 'Diffie-Hellman-over-Bitcoin' protocol to derive their secret k and their partner's estimated public key $\widehat{Q}$. However, a subtle difference requires each user to compare their real public key Q with the estimation of their own key $\widehat{Q}$ to determine if they are equal. If these public keys are different, then the user will use the additive inverse of k as their estimated transaction-specific private key and we will denote this choice between the two keys as $\widehat{k}$. This subtle change will allow both parties to derive the same shared secret $(x_{\text{AB}}, y_{\text{AB}})$ which would be expected in an interactive protocol without exchanging their real y co-ordinates.

Each user generates an ephemeral private key w and computes the corresponding public key $W = wP$. As required in the original YAK paper [12], each user must also construct a zero knowledge proof to prove possession of the ephemeral private key w. These zero knowledge proofs can be sent over an insecure communication channel to their partners. Here, we will use the same Schnorr signature as in [12] to realise the ZKP. Details of the Schnorr ZKP are summarised in Algorithms 2 and 3. The definition of the Schnorr ZKP includes a unique signer identity ID, which prevents an attacker replaying the ZKP back to the signer herself [12]. In our case, we can simply use the unique r value from the user's ECDSA signature (r, s) in the associated Bitcoin transaction T as the user's identity.

Once the ZKPs have been verified, each user will derive $(x_{\text{AB}}, y_{\text{AB}})$ using their secret $w, \widehat{k}$, public value W and their partners' estimated transaction-specific public key $\widehat{Q}$. It should be easy to verify that although the shared secret has four different combinations $(\pm\widehat{k}_{\text{A}} + w_{\text{A}})(\pm\widehat{k}_{\text{B}} + w_{\text{B}})P$, the secret key derived between

Alice and Bob will always be identical (due to each participant predicting the estimated public key $\widehat{Q}$ that their partner will choose).

4 Security Analysis

Our protocols are based on reusing the signature-specific random value k in ECDSA as the transaction-specific secret on which the authenticated key exchange protocol is based. Hence, the security of both the ECDSA signature and the key exchange protocols needs to be analysed to make sure the reusing of k is sound in terms of security.

Algorithm 3. Schnorr Zero Knowledge Proof Verification Algorithm

Input: Domain parameters $D = (q, P, n, \text{Curve})$, signer identity ID, public value W, Schnorr zero knowledge proof values (V, z)
Output: Valid or invalid

1: Perform public key validation for W [13]
2: Compute partners $h = H(D, W, V, ID)$
3: Return $V \stackrel{?}{=} zP + hW \mod n$

For the AKE protocols, following the security analysis of YAK [12], we consider three security requirements, informally defined in the following:

- **Private key security:** The adversary is unable to gain any *extra*[2] information about the private key of an honest party by eavesdropping her communication with other parties, changing messages sent to her, or even participating in an AKE protocol with her.
- **Full forward secrecy:** The adversary is unable to determine the shared secret of an eavesdropped AKE session in the past between a pair of honest parties, even if their private keys are leaked subsequently.
- **Session key security:** The adversary is unable to determine the shared secret between two honest parties by eavesdropping their communication or changing their messages.

Note that in our security arguments we consider the security of *shared secrets* (x_{AB} in Figs. 2 and 3), as opposed to that of the subsequently calculated shared *session keys* (κ in the same figures). We henceforth denote the shared secret by K, i.e., $\kappa = \text{KDF}(K)$. We require the shared secret to be hard to determine for the adversary in the full forward secrecy and session key security requirements. A good key derivation function (KDF) derives from such a shared secret a session key which is indistinguishable from random. Our security proofs can be easily adapted to prove indistinguishability based on the decisional rather than computational Diffie-Hellman assumption.

[2] By "extra" information, we mean information other than what is derivable from the honest party's already available public key.

For ECDSA signature, we require that it remains unforgeable against chosen-message attacks despite the randomness k being reused in subsequent AKE protocols. Although ECDSA has withstood major cryptanalysis, the security of ECDSA has only been proven under non-standard assumptions or assuming modifications (see [28] for a survey of these results). In our analysis, we consider extra information available to an attacker as a result of k being reused in AKE protocols, and show that it does not degrade the security of ECDSA.

We assume ECDSA to be a (non-interactive honest-verifier) zero-knowledge proof of knowledge of the private key d. This is a reasonable assumption in the random oracle model which follows the work of Malone-Lee and Smart [16][3]. In practice, people accept bitcoin transactions only when the ECDSA signatures are verified successfully[4]. Verifying the ECDSA signature is tantamount to verifying the knowledge of the ECDSA private key d that should only be held by the legitimate bitcoin user.

We also note that given an ECDSA message-signature pair, $m, (r, s)$, knowledge of the private key d is equivalent to knowledge of the randomness k since given either the other can be calculated from $sk = H(m) + dr \bmod n$.

4.1 Security of Diffie-Hellman-over-Bitcoin

This protocol is an Elliptic Curve Diffie-Hellman key exchange and the public values are bound to two transactions in the Blockchain. Private key security considers a malicious active adversary "Mallory", and session key security considers an eavesdropper adversary "Eve". The protocol does not provide full forward secrecy. We will provide a sketch of the proof of security for each property in the following. In each proof, we follow the same approach as in [12] to assume an extreme adversary, who has all the powers except those that would allow the attacker to trivially break any key exchange protocol.

Theorem 1 (Private Key Security). *Diffie-Hellman-over-Bitcoin provides private key security under the assumption that ECDSA signature is a zero knowledge proof of knowledge of the ECDSA secret key.*

[3] Note that the results apply to a slightly modified version of ECDSA in which $e = H(r|m)$ where | denotes concatenation. Although the Bitcoin Core implementation is based on the original ECDSA standard, the above modification is included in more recent standards of ECDSA such as ISO/IEC 14888 [1]. Furthermore, as another option for signing, the Bitcoin community is considering including Schnorr signature [2], which is proven to be a zero-knowledge proof of knowledge of the private key.

[4] A bug in the Bitcoin implementation for the SIGHASH_SINGLE flag allows the message that is signed to authorise the transaction to be 1 instead of the hash of the transaction [8]. This bug is not likely to be fixed in the near-future as it is consensus-critical code. To address this bug, we assume that an implementation of our protocol properly checks that the message signed is a hash of a valid transaction as published on the Blockchain rather than 1.

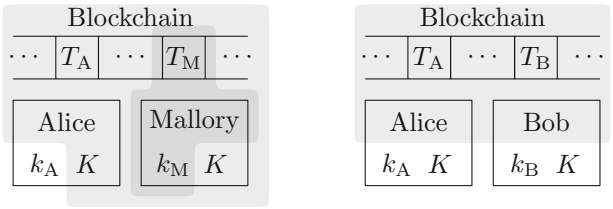

Fig. 4. Security of Diffie-Hellman-over-Bitcoin. Light grey denotes what the adversary (Mallory in (i), Eve in (ii)) *knows*. Dark grey denotes what the adversary (Mallory) *chooses*.

Proof (sketch). The goal of an adversary Mallory is to be able to gain some *extra* information on Alice's transaction-specific private key k_A through the following attack. Mallory is given the public parameters of the system and access to the Blockchain which includes Alice's transaction T_A, then she provides a transaction T_M which is included in the Blockchain, then she carries out a Diffie-Hellman-over-Bitcoin protocol with Alice (which is non-interactive), and eventually is able to calculate the shared secret K. The attack is depicted in Fig. 4(i). Alice's ECDSA signature in T_A is assumed to be zero knowledge and hence does not reveal any information about her private key. Furthermore, since Mallory's transaction T_M includes an ECDSA signature by her, and ECDSA signature is a proof of knowledge of Mallory's ECDSA secret key d_M, Mallory must know d_M, and hence k_M. Hence, Mallory does not gain any extra knowledge from calculating K, since knowledge of k_M and Alice's public key enables her to simulate K on her own. □

Theorem 2 (Session Key Security). *Diffie-Hellman-over-Bitcoin provides session key security based on the computational Diffie-Hellman assumption under the assumption that ECDSA signature is a zero knowledge proof of knowledge of the ECDSA secret key.*

Proof (sketch). Assume there is a successful adversary Eve that is able to calculate the shared secret K for a key exchange between two honest parties Alice and Bob, without knowing either Alice or Bob's transaction-specific secret keys, k_A or k_B. The attack is depicted in Fig. 4(ii). Note that since the protocol is non-interactive, the adversary is reduced to a passive adversary. A successful attack would contradict the computational Diffie-Hellman (CDH) assumption since given an instance of the CDH problem $(P, \alpha P, \beta P)$, one is able to leverage Eve and solve the CDH problem by setting up Alice and Bob's transaction-specific secrets as $k_A = \alpha$ and $k_B = \beta$, which results in $K = \alpha\beta P$. A successful Eve implies that CDH can be solved efficiently. □

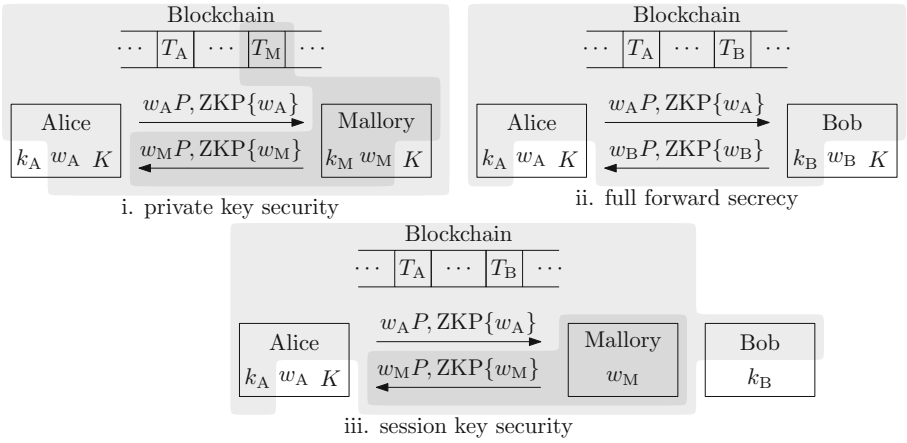

Fig. 5. Security of YAK-over-Bitcoin. Light grey denotes what the adversary (Mallory in (i) and (iii), Eve in (ii)) *knows*. Dark grey denotes what the adversary (Mallory) *chooses*.

4.2 Security of YAK-over-Bitcoin

This protocol is an Elliptic Curve YAK key exchange and the public values are bound to two transactions in the Blockchain. Private key security and session key security consider a malicious active adversary "Mallory", and full forward secrecy considers an eavesdropper adversary "Eve". Similar as before, we assume an extreme adversary who has all the powers except those that would trivially allow the attacker to break any key exchange protocol. Under this assumption, we provide a sketch of the proof of security for each property in the following.

Theorem 3 (Private Key Security). *YAK-over-Bitcoin provides private key security under the assumption that ECDSA signature is a zero knowledge proof of knowledge of the ECDSA secret key.*

Proof (sketch). The goal of an adversary Mallory is to be able to gain some *extra* information on Alice's transaction-specific private key k_A through the following attack. Mallory is given the public parameters of the system and access to the Blockchain which includes Alice's transaction T_A, then she provides a transaction T_M which is included in the Blockchain, then she carries out a YAK-over-Bitcoin protocol with Alice, in which Alice sends the message $(w_AP, \mathrm{ZKP}\{w_A\})$ and Mallory sends the message $(w_MP, \mathrm{ZKP}\{w_M\})$. Alice's ephemeral secret w_A is also assumed to be leaked to Mallory. The attack is depicted in Fig. 5(i). Alice's ECDSA signature in T_A is assumed to be zero knowledge and hence does not reveal any information about her private key. Furthermore, since the ECDSA signature in Mallory's transaction and her message in the protocol are proofs of knowledge of d_M (equivalently k_M) and w_M, respectively, Mallory must know both k_M and w_M. Note that she receives $(w_AP, \mathrm{ZKP}\{w_A\})$ and w_A and hence will be able to calculate $K = (k_M + w_M)(k_AP + w_AP)$. Hence, Mallory does not

gain any extra knowledge from the values she receives, since w_A is independent of k_A and knowledge of w_A, k_M, and w_M enables Mallory to simulate all the values she receives. □

Theorem 4 (Full Forward Secrecy). *YAK-over-Bitcoin provides full forward secrecy based on the computational Diffie-Hellman assumption.*

Proof (sketch). Assume there is a successful adversary Eve that is able to calculate the shared secret K for a previous key exchange between two honest parties Alice and Bob through the following attack. Both Alice and Bob's transaction-specific secret keys k_A and k_B are assumed to be leaked to Eve. Eve is also assumed to have access to all the protocol messages exchanged between Alice and Bob, as well as the Blockchain of course. The attack is depicted in Fig. 5(ii). Given an instance of the CDH problem $(P, \alpha P, \beta P)$ one is able to leverage Eve and solve the problem as follows. The protocol is set up with the ephemeral secret values $w_A = \alpha$ and $w_B = \beta$ and all other parameters as per the protocol description. When Eve calculates K, the value $S = K - k_A k_B P - k_A(\beta P) - k_B(\alpha P)$ is calculated and returned as the solution to the CDH problem. Note that since $K = (k_A + w_A)(k_B + w_B)P$, we have $S = \alpha\beta P$. A successful Eve implies that CDH can be solved efficiently. □

Theorem 5 (Session Key Security). *YAK-over-Bitcoin provides session key security based on the computational Diffie-Hellman assumption under the assumption that ECDSA signature is a zero knowledge proof of knowledge of the ECDSA secret key.*

Proof (sketch). Assume there is a successful adversary Mallory that is able to calculate the shared secret K for a key exchange between two honest parties Alice and Bob through the following attack by impersonating Bob to Alice. Alice believes she is interacting with Bob, whereas in reality she is interacting with an impersonator Mallory who replaces Bob's message in the protocol with her own $(w_M P, \text{ZKP}\{w_M\})$. Alice's transaction-specific secret key k_A is assumed to be leaked to Mallory as well. However, Mallory does not know Bob's transaction-specific secret key k_B. The attack is depicted in Fig. 5(iii). Given an instance of the CDH problem $(P, \alpha P, \beta P)$ one is able to leverage Mallory and solve the problem as follows. The protocol is set up with Alice's ephemeral secret $w_A = \alpha$ and Bob's transaction-specific secret $k_B = \beta$ and all other parameters as per the protocol description. When Mallory calculates K, the value $S = K - k_A w_B P - w_A(\beta P) - w_B(\alpha P)$ is calculated and returned as the solution to the CDH problem. Note that since $K = (k_A + w_A)(k_B + w_B)P$, we have $S = \alpha\beta P$. A successful Mallory implies that CDH can be solved efficiently. □

4.3 Security of ECDSA Signatures

Diffie-Hellman-over-Bitcoin is a non-interactive protocol and the protocol participants do not send any messages to each other that would potentially have an impact on the security of ECDSA signatures.

In 'YAK-over-Bitcoin', the messages that the protocol participants send each other include information about their ephemeral keys w_A and w_B only, which are chosen independently of all the secret values related to the ECDSA signatures in T_A and T_B. As shown in Theorem 1 in Sect. 4.2, the protocol does not reveal any information about the static private key (i.e., k), and hence not any information about the ECDSA private key (i.e., d) since the two values are linearly related. One can compute d from k, or vice versa. The key element in the proof of Theorem 3 is that each party is required to demonstrate knowledge of both the static and ephemeral keys. This also explains our choice of the YAK protocol, as YAK is the only PKI-based AKE protocol that has the requirement that each party must demonstrate the proof of knowledge for both the static and ephemeral keys (the former is realized by the Proof of Possession at the Certificate Authority registration while the later is achieved by Schnorr Non-interactive ZKP).

Table 2. Time performance for Alice executing Diffie-Hellman-over-Bitcoin and YAK-over-Bitcoin

Step	Description	Time
	Diffie-Hellman-over-Bitcoin	
1-2	Compute k_A and $\widehat{Q}_B$	0.08 ms
3	Compute shared secret K_{AB} and key κ_{AB}	0.51 ms
	Total:	0.59 ms
	YAK-over-Bitcoin	
1-4	Compute $k_A, Q_A, \widehat{Q}_A$ and $\widehat{Q}_B$	0.53 ms
5	Compute w_A, W_A and ZKP$\{w_A\}$	0.90 ms
6	Verify Bob's ZKP$\{w_B\}$	0.69 ms
7	Compute shared secret K_{AB} and key κ_{AB}	0.43 ms
	Total:	2.55 ms

5 Implementation

Our implementation is a modification of the Bitcoin Core client and is considered a proof of concept. We have included three new remote procedure commands (RPC) that will allow the client to perform a non-interactive Diffie-Hellman key exchange, generate a zero knowledge proof to be shared with their partner and verify a partner's zero knowledge proof before revealing the shared secret. Our modified implementation was executed using the -txindex parameter which allows us to query the Blockchain and retrieve the raw transaction data.

Two transactions were created using a non-modified implementation on the 10th December, 2013 to allow us to test our key exchange on the real network. All tests were carried out a MacBook Pro mid-2012 running OS X 10.9.1 with 2.3GHz Intel Core i7 and 4 cores and 16 GB DDR3 RAM. Each protocol is executed 100 times from Alice's perspective and the average times are reported.

5.1 Time Analysis

Preliminary steps for both protocols involve fetching the transactions from the Blockchain 0.04 ms and retrieving the signatures (r, s) stored in the transaction 0.08 ms. Overall, these steps on average require 0.12 ms.

This 'Diffie-Hellman-over-Bitcoin' protocol is non-interactive as participants are not required to exchange information before deriving the shared secret. Table 2 demonstrates an average time of 0.08 ms to derive Alice's transaction-specific private key k_{A} and Bob's estimated public key $\widehat{Q}_{\mathrm{B}}$ and 0.051 ms to compute the shared key κ_{AB}. Overall, on average the protocol executes in 0.59 ms which is reasonable for real-life use.

The 'YAK-over-Bitcoin' protocol is interactive as it requires each party to send an ephemeral public key together with a non-interactive Schnorr ZKP to prove the knowledge of the ephemeral private key. Table 2 shows that computing and verifying zero knowledge proofs is the most time-consuming operation. However, a total execution time of 2.55 ms is still reasonable for practical applications.

5.2 Note About Domain Parameters

Our investigation highlighted that $q > n$ as seen in [6] which could obscure the relationship between k and r as the x co-ordinate can wrap around n. However, the probability that this may occur can be calculated as $(q - n)/q \approx 4 \times 10^{-39}$ and is unlikely to occur in practice. However, in the rare chance that this does happen then it is easily resolved by $r' = r + n$. This does not require any modification to the underlying signature code as it is simply an addition of the publicly available r with the modulus n. Once resolved, both parties can continue with the protocol. For reference, q and n are defined below:

```
q=FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
n=FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
```

6 Conclusion

In this paper, we have demonstrated transaction authentication by using the digital signatures stored in Bitcoin transactions to bootstrap key exchange. We proposed two protocols to allow for interactive and non-interactive key exchange, the latter offering an additional property of forward-secrecy. We encourage the community to try our proof-of-concept implementation and to take advantage of this new form of authentication to enable end-to-end secure communication between Bitcoin users.

Acknowledgements. The second, third and fourth authors are supported by the European Research Council (ERC) Starting Grant (No. 306994). We also thank Greg Maxwell for bringing the SIGHASH_SINGLE implementation bug to our attention.

References

1. ISO/IEC 14888: Information technology - Security techniques - Digital signatures with appendix (2008)
2. Andersen, G.: Conversation about OP_SCHNORRVERIFY. Freenode IRC bitcoin-wizards, October 2014. https://botbot.me/freenode/bitcoin-wizards/
3. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 34–51. Springer, Heidelberg (2013)
4. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to better — how to make bitcoin a better currency. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 399–414. Springer, Heidelberg (2012)
5. BBC: New Paypal partnership enables limited Bitcoin payments (2015). http://www.bbc.co.uk/news/technology-29341886. Accessed 06 January 2015
6. Research, Certicom: SEC 2: Recommended Elliptic Curve Domain Parameters. Standards for Efficient Cryptography Group, September 2000
7. Clark, J., Essex, A.: CommitCoin: carbon dating commitments with bitcoin. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 390–398. Springer, Heidelberg (2012)
8. Corallo, M.: [Bitcoin-development] Warning to rawtx creators: bug in SIGHASH_SINGLE (2012). http://sourceforge.net/p/bitcoin/mailman/message/29699385/. Accessed 16 September 2015
9. Dell: Were Now Accepting Bitcoin on Dell.com (2015). http://en.community.dell.com/dell-blogs/direct2dell/b/direct2dell/archive/2014/07/18/we-re-now-accepting-bitcoin-on-dell-com. Accessed January 06 2015
10. Eyal, I., Sirer, E.G.: Majority is not enough: bitcoin mining is vulnerable (2013). arXiv preprint arXiv:1311.0243
11. Hankerson, D., Vanstone, S., Menezes, A.: Guide to Elliptic Curve Cryptography. Springer Professional Computing. Springer, New York (2004)
12. Hao, F.: On robust key agreement based on public key authentication. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 383–390. Springer, Heidelberg (2010)
13. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Secur. **1**(1), 36–63 (2001)
14. Karame, G.O., Androulaki, E., Capkun, S.: Double-spending fast payments in bitcoin. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 906–917. ACM (2012)
15. Lo, S., Wang, J.: Bitcoin as money? current policy and perspectives, September 2014
16. Malone-Lee, J., Smart, N.P.: Modifications of ECDSA. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 1–12. Springer, Heidelberg (2003)
17. Maurer, B., Nelms, T., Swartz, L.: When perhaps the real problem is money itself!: the practical materiality of Bitcoin. Soc. Semiot. **23**(2), 261–277 (2013)
18. Merrill, N.: The Calyx institute: privacy by design for everyone (2015). https://www.calyxinstitute.org/support-us/donate-via-bitcoin. Accessed January 06 2015
19. Miers, I., Garman, C., Green, M., Rubin, A.: Zerocoin: anonymous distributed E-cash from Bitcoin. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 397–411. IEEE (2013)
20. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)

21. Mozilla: Help protect the open Web (2015). https://sendto.mozilla.org/page/content/give-bitcoin/. Accessed January 06 2015
22. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
23. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: 2011 IEEE Third International Conference on Social Computing (socialcom) Privacy, Security, Risk and Trust (Passat), pp. 1318–1326, October 2011
24. Rizzo, P.: Expedia exec says bitcoin spending has exceeded estimates (2015). http://www.coindesk.com/expedia-exec-bitcoin-payments-have-exceeded-estimates/. Accessed January 06 2015
25. Robleh, A., Barrdear, J., Clews, R., Southgate, J.: The economics of digital currencies. Q. Bull. **54**, Q3 (2014)
26. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 6–24. Springer, Heidelberg (2013)
27. Tor: Make A Donation. 2015. https://www.torproject.org/donate/donate.html.en. Accessed January 06 2015
28. Vaudenay, S.: The security of DSA and ECDSA. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 309–323. Springer, Heidelberg (2002)
29. Woo, D., Gordon, I., Iaralov, V.: Bitcoin: a first assessment. Bank of America Merrill Lynch, December 2013