BIP 70 and the Future of Bitcoin Payments

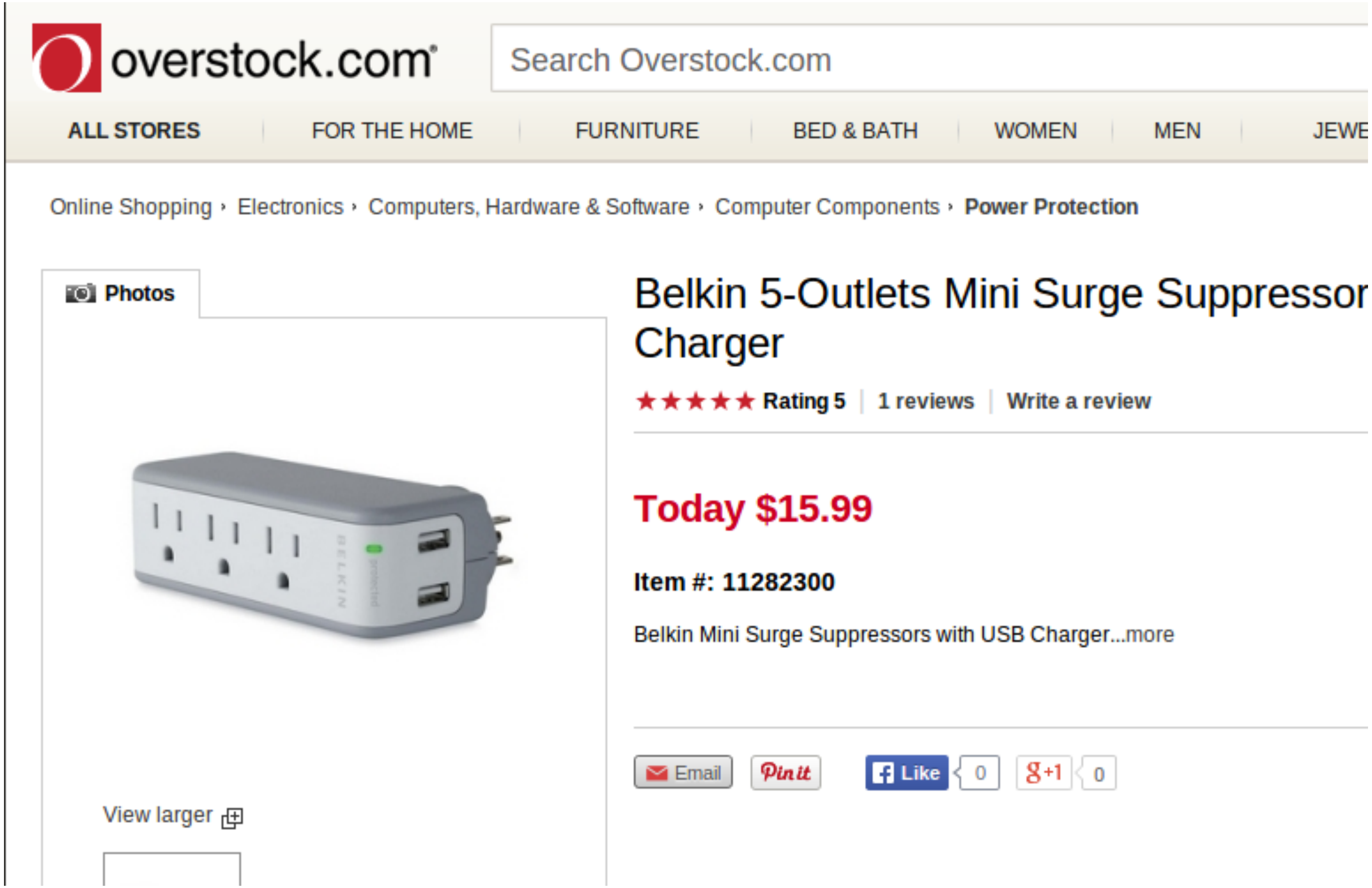
Kevin Greene

1. Paying with bitcoin today

2. The Payment Protocol (BIP 70)

3. Uses, ideas and future extensions

4. The Payment Protocol in bitcoinj


overstock.com

Search Overstock.com

ALL STORES | FOR THE HOME | FURNITURE | BED & BATH | WOMEN | MEN | JEWE

Online Shopping › Electronics › Computers, Hardware & Software › Computer Components › **Power Protection**

Photos

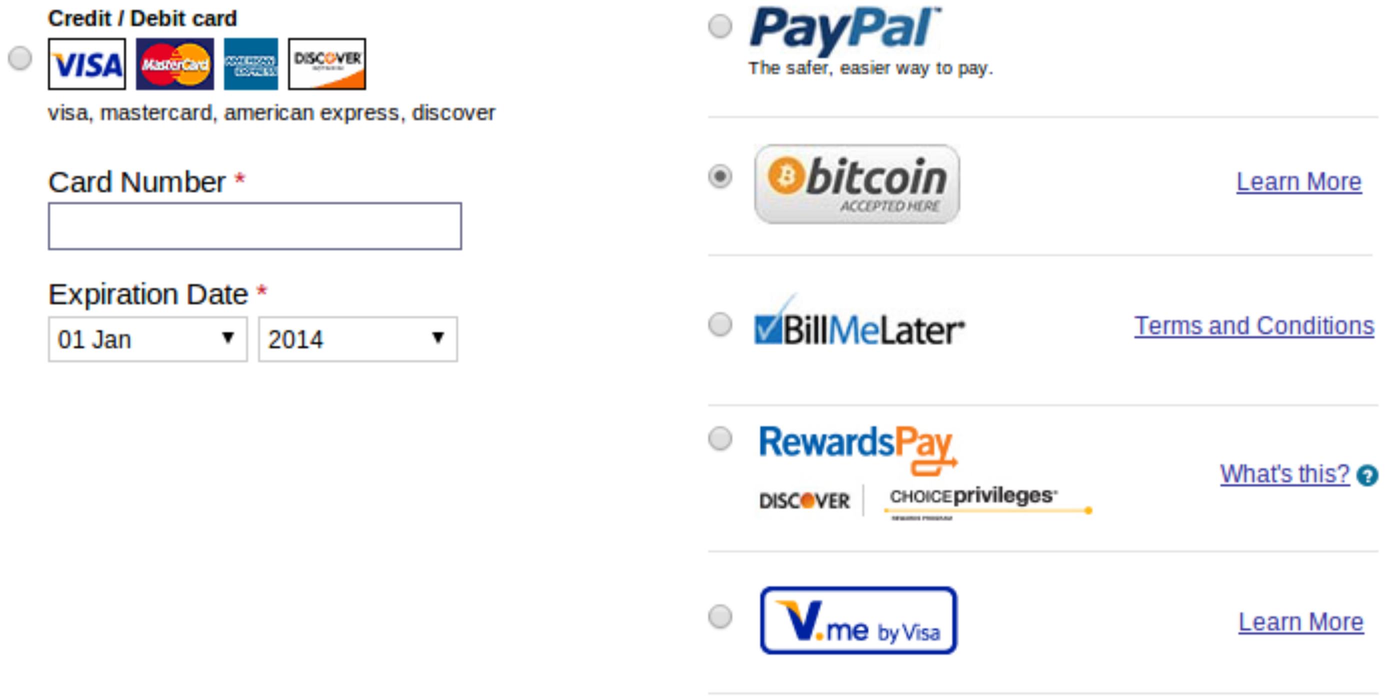
Belkin 5-Outlets Mini Surge Suppressor Charger

★★★★★ **Rating 5** | **1 reviews** | **Write a review**

Today $15.99

Item #: 11282300

Belkin Mini Surge Suppressors with USB Charger...more



Credit / Debit card

visa, mastercard, american express, discover

Card Number *

Expiration Date *

01 Jan ▼ 2014 ▼

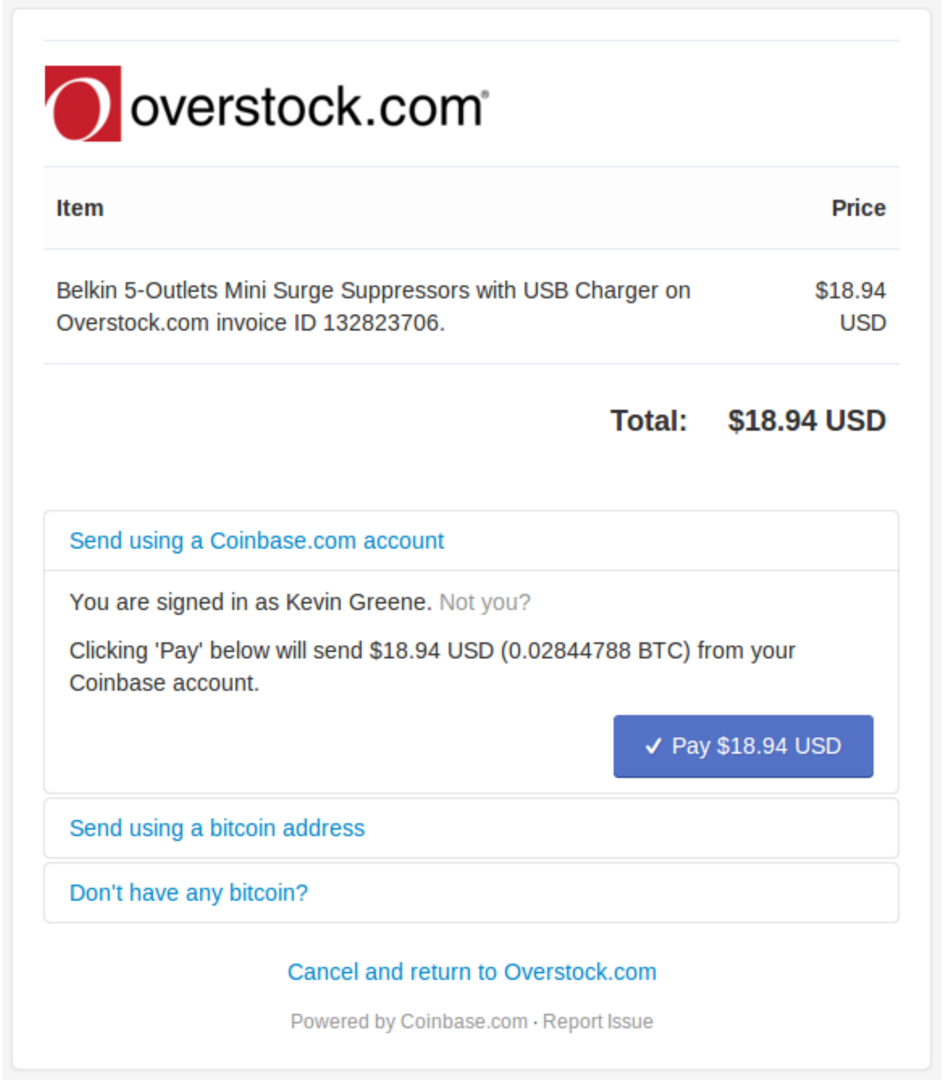
PayPal — The safer, easier way to pay.

[✓ bitcoin ACCEPTED HERE] — Learn More

BillMeLater — Terms and Conditions

RewardsPay — DISCOVER | CHOICEprivileges — What's this?

V.me by Visa — Learn More



overstock.com

Item	Price
Belkin 5-Outlets Mini Surge Suppressors with USB Charger on Overstock.com invoice ID 132823706.	$18.94 USD

Total: $18.94 USD

Send using a Coinbase.com account

You are signed in as Kevin Greene. Not you?

Clicking 'Pay' below will send $18.94 USD (0.02844788 BTC) from your Coinbase account.

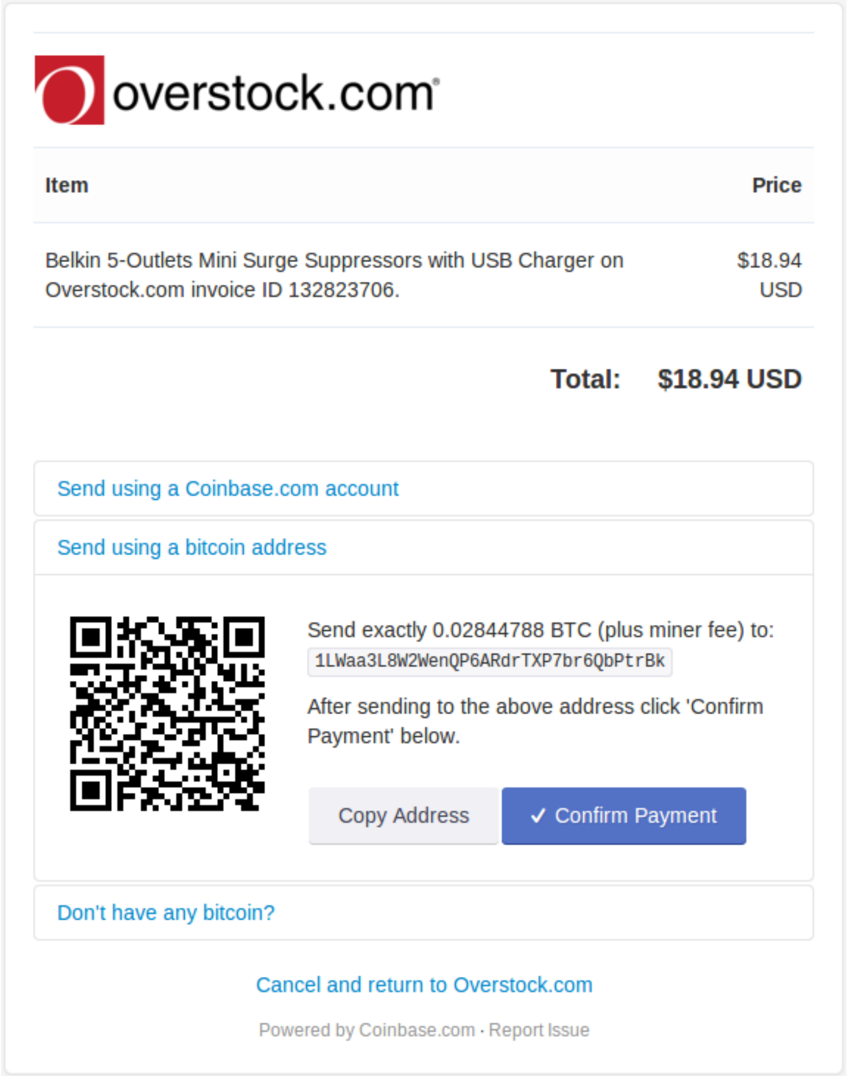
✓ Pay $18.94 USD

Send using a bitcoin address

Don't have any bitcoin?

Cancel and return to Overstock.com

Powered by Coinbase.com · Report Issue



overstock.com

Item	Price
Belkin 5-Outlets Mini Surge Suppressors with USB Charger on Overstock.com invoice ID 132823706.	$18.94 USD

Total: $18.94 USD

Send using a Coinbase.com account

Send using a bitcoin address

Send exactly 0.02844788 BTC (plus miner fee) to:

`1LWaa3L8W2WenQP6ARdrTXP7br6QbPtrBk`

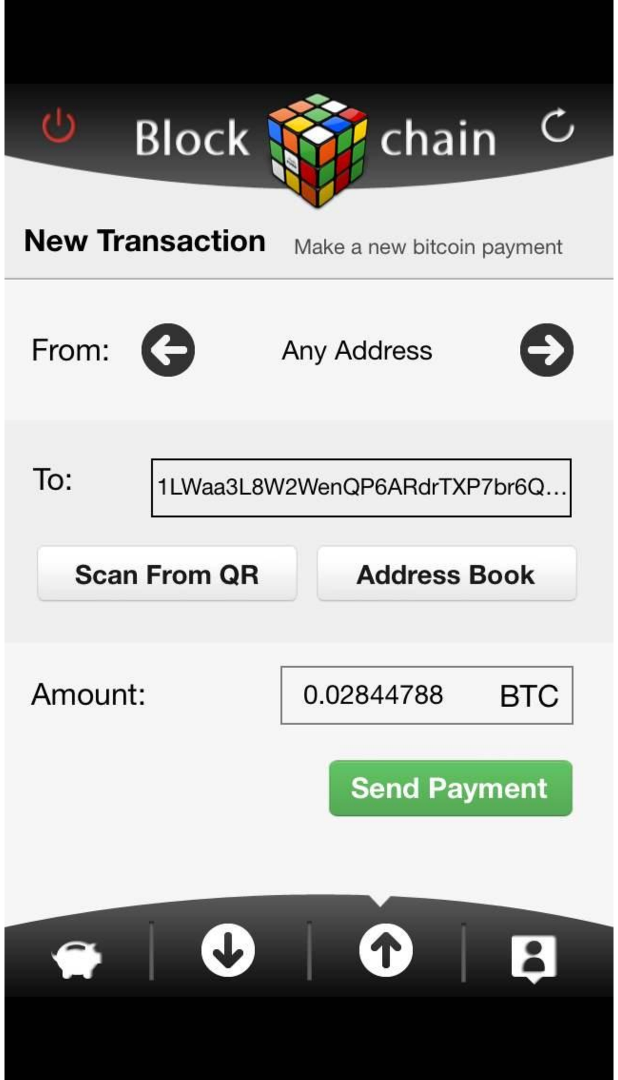
After sending to the above address click 'Confirm Payment' below.

Copy Address ✓ Confirm Payment

Don't have any bitcoin?

Cancel and return to Overstock.com

Powered by Coinbase.com · Report Issue

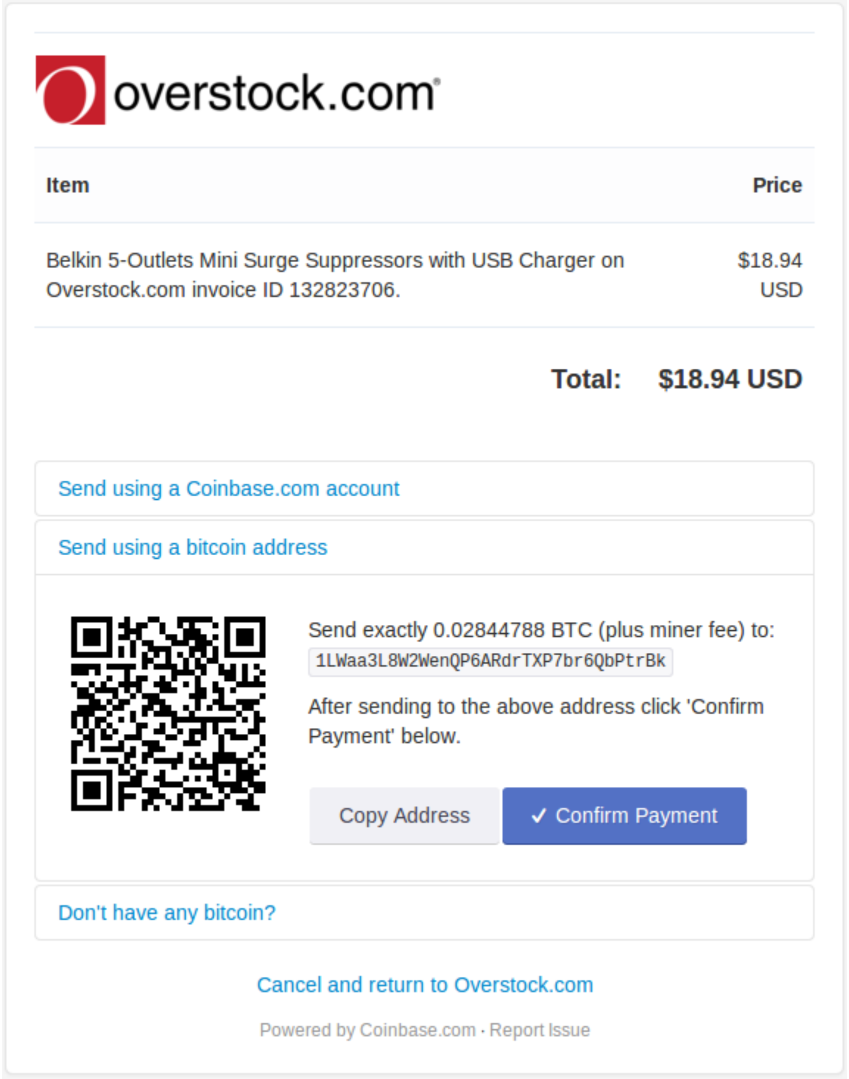


New Transaction Make a new bitcoin payment

From: Any Address

To: 1LWaa3L8W2WenQP6ARdrTXP7br6Q…

Amount: 0.02844788 BTC



overstock.com

Item	Price
Belkin 5-Outlets Mini Surge Suppressors with USB Charger on Overstock.com invoice ID 132823706.	$18.94 USD

Total: $18.94 USD

Send using a Coinbase.com account

Send using a bitcoin address

Send exactly 0.02844788 BTC (plus miner fee) to:

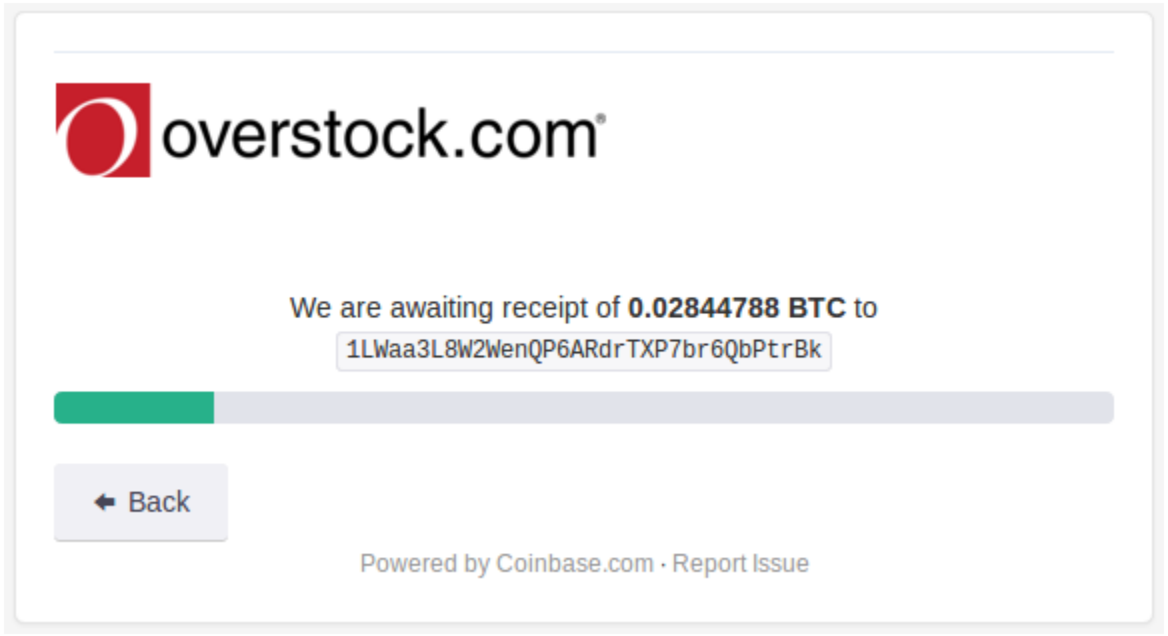
`1LWaa3L8W2WenQP6ARdrTXP7br6QbPtrBk`

After sending to the above address click 'Confirm Payment' below.

Don't have any bitcoin?

Cancel and return to Overstock.com

Powered by Coinbase.com · Report Issue

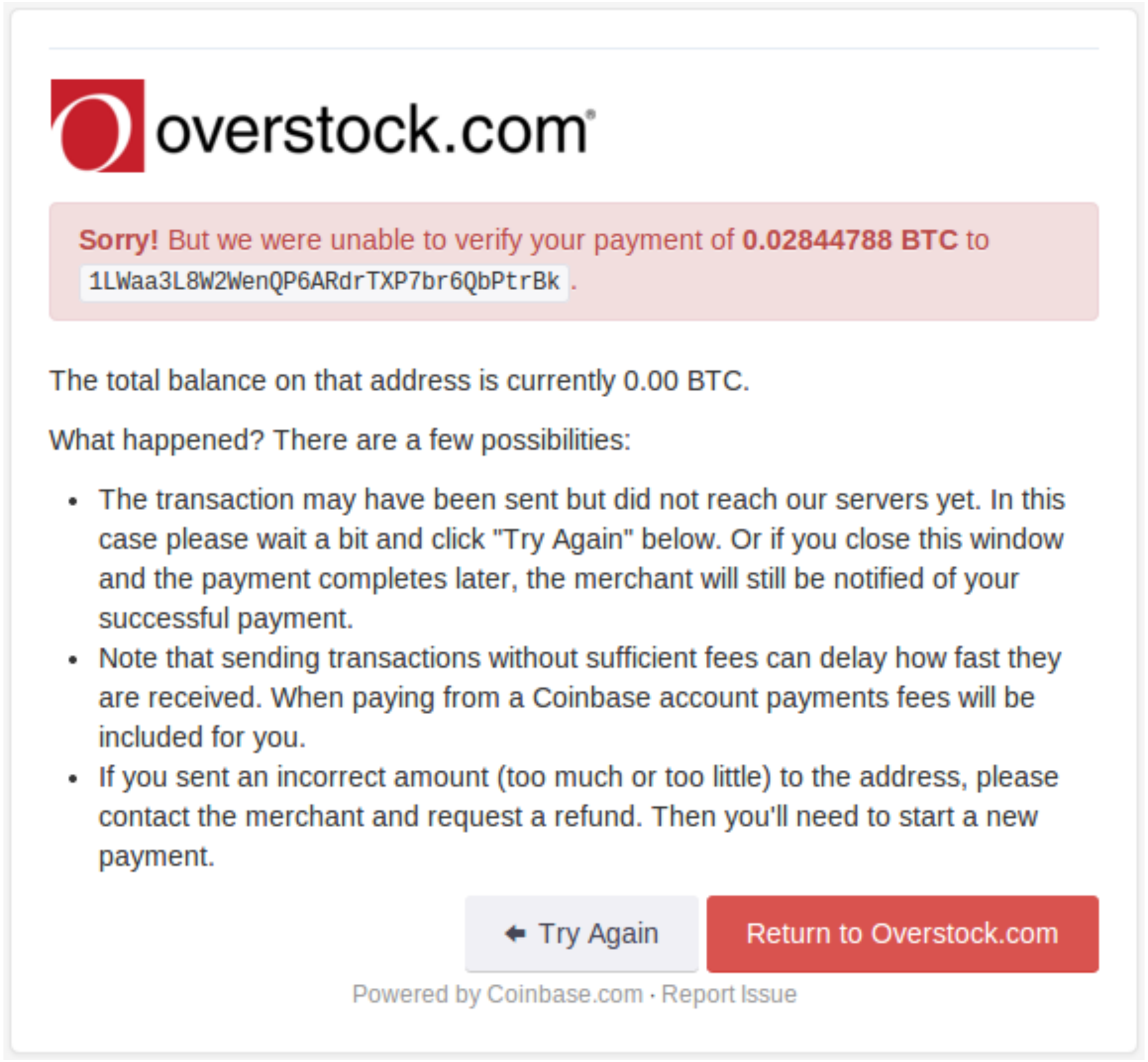


overstock.com

We are awaiting receipt of **0.02844788 BTC** to

`1LWaa3L8W2WenQP6ARdrTXP7br6QbPtrBk`

Powered by Coinbase.com · Report Issue



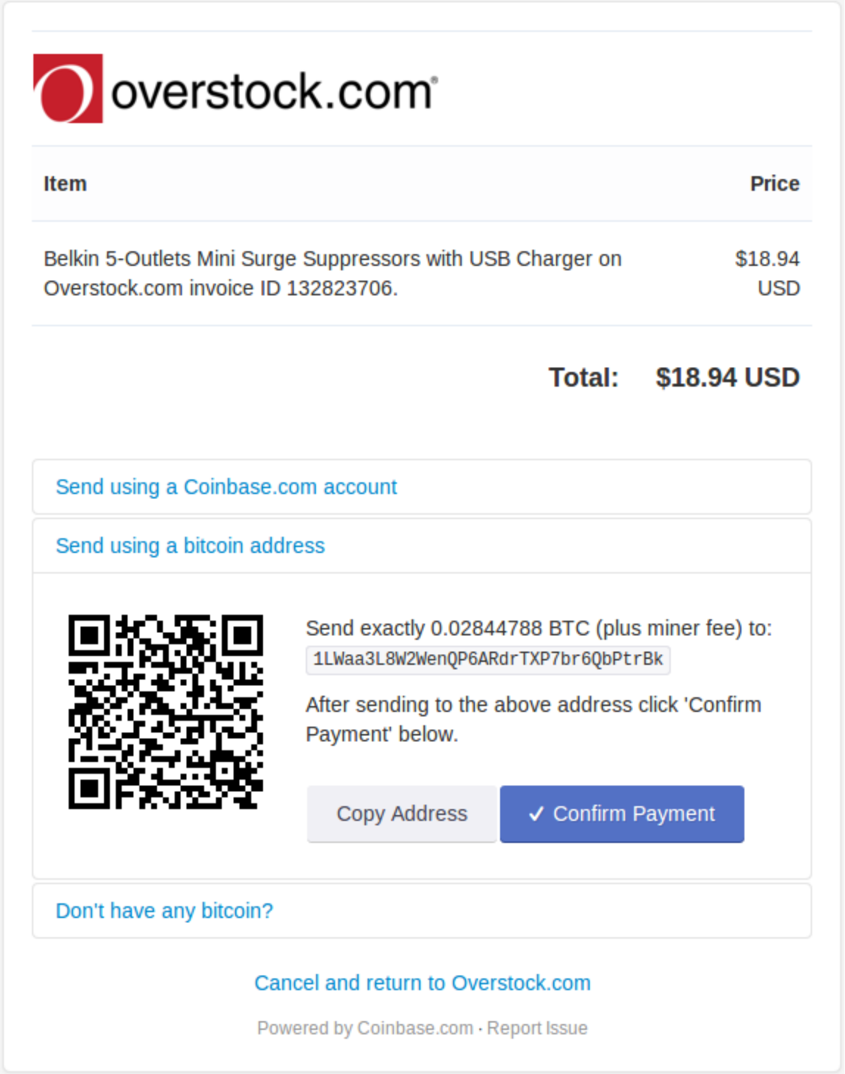
overstock.com

Sorry! But we were unable to verify your payment of **0.02844788 BTC** to `1LWaa3L8W2WenQP6ARdrTXP7br6QbPtrBk`.

The total balance on that address is currently 0.00 BTC.

What happened? There are a few possibilities:

- The transaction may have been sent but did not reach our servers yet. In this case please wait a bit and click "Try Again" below. Or if you close this window and the payment completes later, the merchant will still be notified of your successful payment.
- Note that sending transactions without sufficient fees can delay how fast they are received. When paying from a Coinbase account payments fees will be included for you.
- If you sent an incorrect amount (too much or too little) to the address, please contact the merchant and request a refund. Then you'll need to start a new payment.



overstock.com

Item	Price
Belkin 5-Outlets Mini Surge Suppressors with USB Charger on Overstock.com invoice ID 132823706.	$18.94 USD

Total: $18.94 USD

Send using a Coinbase.com account

Send using a bitcoin address

Send exactly 0.02844788 BTC (plus miner fee) to:

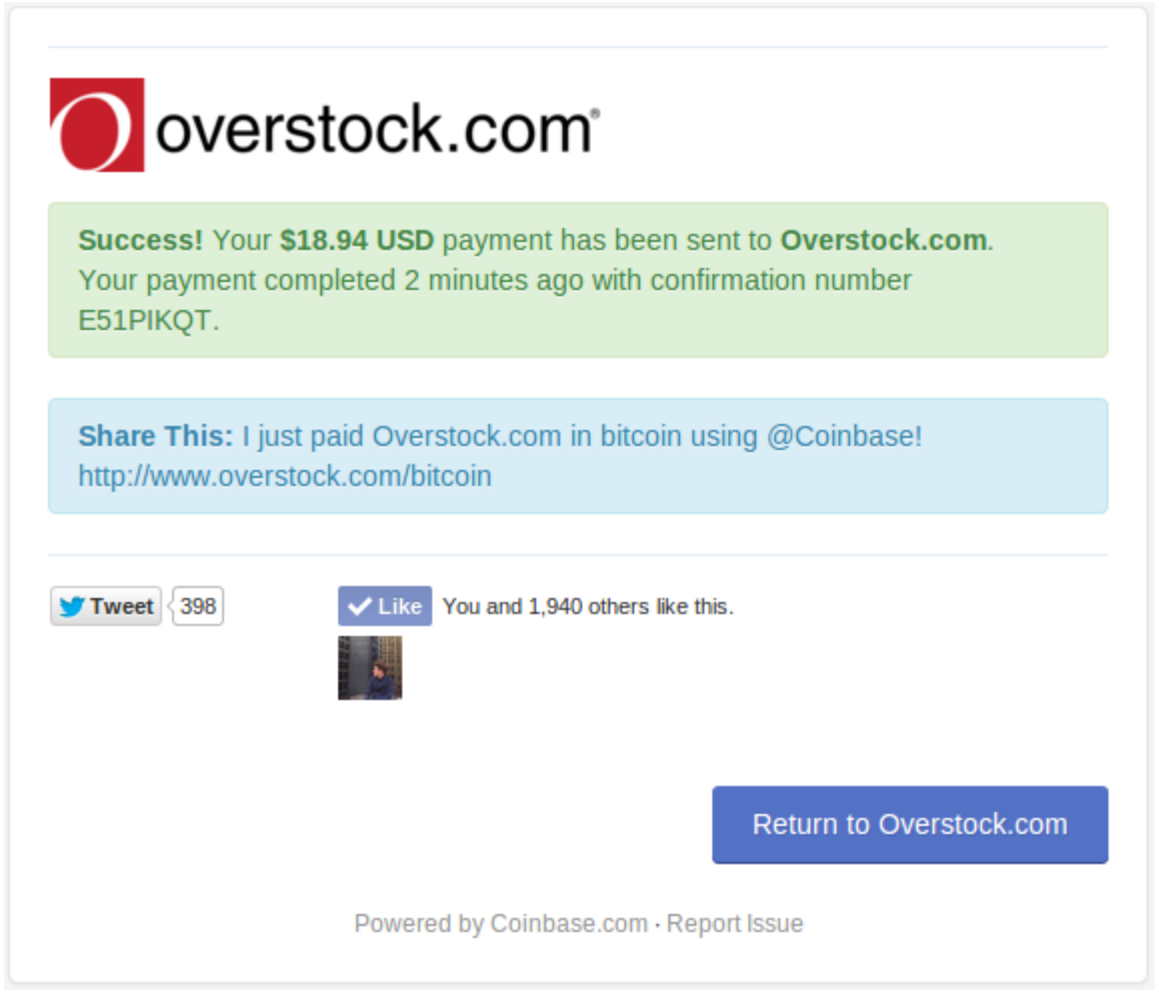
`1LWaa3L8W2WenQP6ARdrTXP7br6QbPtrBk`

After sending to the above address click 'Confirm Payment' below.

Don't have any bitcoin?

Cancel and return to Overstock.com

Powered by Coinbase.com · Report Issue



overstock.com®

Success! Your **$18.94 USD** payment has been sent to **Overstock.com**. Your payment completed 2 minutes ago with confirmation number E51PIKQT.

Share This: I just paid Overstock.com in bitcoin using @Coinbase! http://www.overstock.com/bitcoin

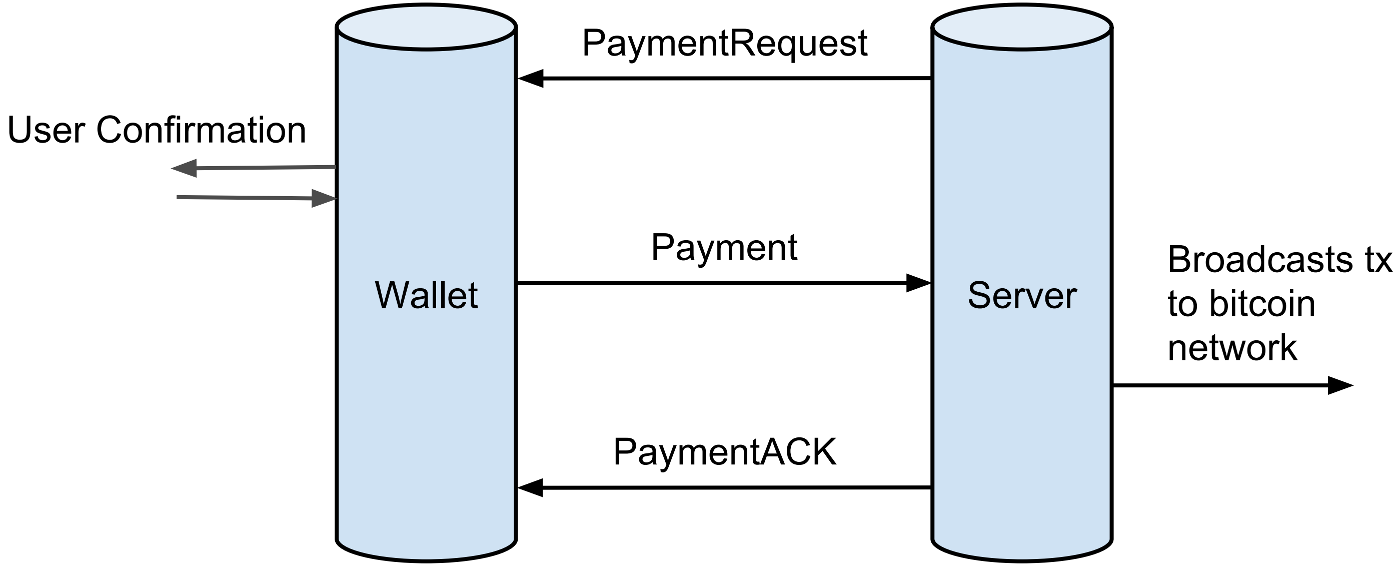
Tweet 398

Like You and 1,940 others like this.

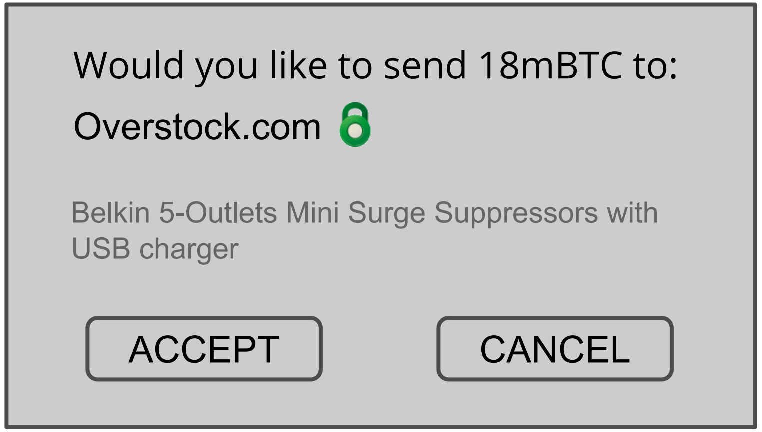
Return to Overstock.com

Powered by Coinbase.com · Report Issue

1. Paying with bitcoin today
2. **The Payment Protocol (BIP 70)**
3. Uses, ideas and future extensions
4. The Payment Protocol in bitcoinj

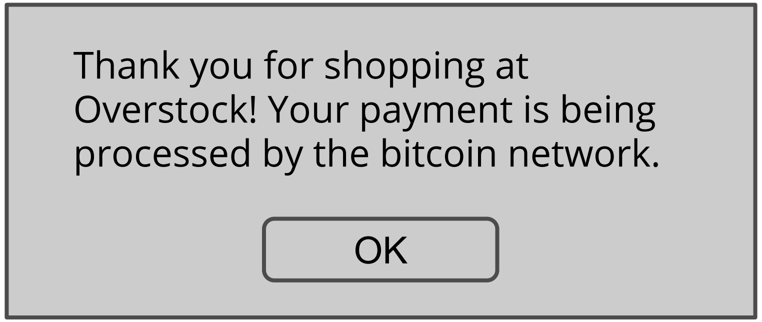

PaymentRequest
User Confirmation
Payment
Wallet
Server
Broadcasts tx to bitcoin network
PaymentACK

Click here to send 18mBTC to Overstock.com


Would you like to send 18mBTC to:

Overstock.com

Belkin 5-Outlets Mini Surge Suppressors with USB charger


Thank you for shopping at Overstock! Your payment is being processed by the bitcoin network.

Bitcoin URIs

Old (deprecated) URI -- BIP 21:

```
bitcoin:mq7se9wy2egettFxPbmn99cK8v5AFq55Lx?amount=0.11
```

New Payment Protocol URI -- BIP 72:

```
bitcoin:?r=https://bitpay.com/i/X5h6Q9tD6v4i5gj8JX6en9
```

```
message PaymentDetails {
        optional string network = 1 [default = "main"]; // "main" or "test"
        repeated Output outputs = 2;        // Where payment should be sent
        required uint64 time = 3;           // Timestamp; when payment request created
        optional uint64 expires = 4;        // Timestamp; when this request should be considered invalid
        optional string memo = 5;           // Human-readable description of request for the customer
        optional string payment_url = 6;    // URL to send Payment and get PaymentACK
        optional bytes merchant_data = 7;   // Arbitrary data to include in the Payment message
}
message PaymentRequest {
        optional uint32 payment_details_version = 1 [default = 1];
        optional string pki_type = 2 [default = "none"];  // none / x509+sha256 / x509+sha1
        optional bytes pki_data = 3;                      // depends on pki_type
        required bytes serialized_payment_details = 4;    // PaymentDetails
        optional bytes signature = 5;                     // pki-dependent signature
}
message Payment {
        optional bytes merchant_data = 1;  // From PaymentDetails.merchant_data
        repeated bytes transactions = 2;   // Signed transactions that satisfy PaymentDetails.outputs
        repeated Output refund_to = 3;     // Where to send refunds, if a refund is necessary
        optional string memo = 4;          // Human-readable message for the merchant
}
message PaymentACK {
        required Payment payment = 1;      // Payment message that triggered this ACK
        optional string memo = 2;          // human-readable message for customer
}
```

```
message PaymentDetails {
        optional string network = 1 [default = "main"]; // "main" or "test"
        repeated Output outputs = 2;        // Where payment should be sent
        required uint64 time = 3;           // Timestamp; when payment request created
        optional uint64 expires = 4;        // Timestamp; when this request should be considered invalid
        optional string memo = 5;           // Human-readable description of request for the customer
        optional string payment_url = 6;    // URL to send Payment and get PaymentACK
        optional bytes merchant_data = 7;   // Arbitrary data to include in the Payment message
}
message PaymentRequest {
        optional uint32 payment_details_version = 1 [default = 1];
        optional string pki_type = 2 [default = "none"];  // none / x509+sha256 / x509+sha1
        optional bytes pki_data = 3;                      // depends on pki_type
        required bytes serialized_payment_details = 4;    // PaymentDetails
        optional bytes signature = 5;                     // pki-dependent signature
}
message Payment {
        optional bytes merchant_data = 1;  // From PaymentDetails.merchant_data
        repeated bytes transactions = 2;   // Signed transactions that satisfy PaymentDetails.outputs
        repeated Output refund_to = 3;     // Where to send refunds, if a refund is necessary
        optional string memo = 4;          // Human-readable message for the merchant
}
message PaymentACK {
        required Payment payment = 1;      // Payment message that triggered this ACK
        optional string memo = 2;          // human-readable message for customer
}
```

```
message PaymentDetails {
        optional string network = 1 [default = "main"]; // "main" or "test"
        repeated Output outputs = 2;        // Where payment should be sent
        required uint64 time = 3;           // Timestamp; when payment request created
        optional uint64 expires = 4;        // Timestamp; when this request should be considered invalid
        optional string memo = 5;           // Human-readable description of request for the customer
        optional string payment_url = 6;    // URL to send Payment and get PaymentACK
        optional bytes merchant_data = 7;   // Arbitrary data to include in the Payment message
}
message PaymentRequest {
        optional uint32 payment_details_version = 1 [default = 1];
        optional string pki_type = 2 [default = "none"];  // none / x509+sha256 / x509+sha1
        optional bytes pki_data = 3;                      // depends on pki_type
        required bytes serialized_payment_details = 4;    // PaymentDetails
        optional bytes signature = 5;                     // pki-dependent signature
}
message Payment {
        optional bytes merchant_data = 1;  // From PaymentDetails.merchant_data
        repeated bytes transactions = 2;   // Signed transactions that satisfy PaymentDetails.outputs
        repeated Output refund_to = 3;     // Where to send refunds, if a refund is necessary
        optional string memo = 4;          // Human-readable message for the merchant
}
message PaymentACK {
        required Payment payment = 1;      // Payment message that triggered this ACK
        optional string memo = 2;          // human-readable message for customer
}
```

```
message PaymentDetails {
        optional string network = 1 [default = "main"]; // "main" or "test"
        repeated Output outputs = 2;        // Where payment should be sent
        required uint64 time = 3;           // Timestamp; when payment request created
        optional uint64 expires = 4;        // Timestamp; when this request should be considered invalid
        optional string memo = 5;           // Human-readable description of request for the customer
        optional string payment_url = 6;    // URL to send Payment and get PaymentACK
        optional bytes merchant_data = 7;   // Arbitrary data to include in the Payment message
}
message PaymentRequest {
        optional uint32 payment_details_version = 1 [default = 1];
        optional string pki_type = 2 [default = "none"];  // none / x509+sha256 / x509+sha1
        optional bytes pki_data = 3;                      // depends on pki_type
        required bytes serialized_payment_details = 4;    // PaymentDetails
        optional bytes signature = 5;                     // pki-dependent signature
}
message Payment {
        optional bytes merchant_data = 1;  // From PaymentDetails.merchant_data
        repeated bytes transactions = 2;   // Signed transactions that satisfy PaymentDetails.outputs
        repeated Output refund_to = 3;     // Where to send refunds, if a refund is necessary
        optional string memo = 4;          // Human-readable message for the merchant
}
message PaymentACK {
        required Payment payment = 1;      // Payment message that triggered this ACK
        optional string memo = 2;          // human-readable message for customer
}
```

1. Paying with bitcoin today
2. The Payment Protocol (BIP 70)
3. Uses, ideas and future extensions
4. The Payment Protocol in bitcoinj

- Embedded Tipping Links
- Provable Receipts
- Fancy Outputs:
    - Multiple recipients
    - Contracts
- Merge Avoidance
- In-Person Payments Over Bluetooth/NFC
- Recurring Payment Contracts
- Server-side Implementation

1. Paying with bitcoin today
2. The Payment Protocol (BIP 70)
3. Uses, ideas and future extensions
4. The Payment Protocol in bitcoinj

```
public class PaymentSession {
      public static ListenableFuture<PaymentSession> createFromBitcoinUri(final BitcoinURI uri)
            throws PaymentRequestException { … }

      public static ListenableFuture<PaymentSession> createFromUrl(final String url)
            throws PaymentRequestException { … }

      public PaymentSession(Protos.PaymentRequest request) throws PaymentRequestException { … }

      public Wallet.SendRequest getSendRequest() { … }

      public @Nullable ListenableFuture<Ack> sendPayment(List<Transaction> txns,
                                                         @Nullable Address refundAddr,
                                                         @Nullable String memo) { … }

      public @Nullable PkiVerificationData verifyPki() throws PaymentRequestException { … }
}
```

```
String url = QRCodeScanner.scanFromCamera(.....);
ListenableFuture<PaymentSession> future;
if (url.startsWith("http")) {
    // URL may serve either HTML or a payment request depending on how it's
fetched.
    // Try here to get a payment request.
    future = PaymentSession.createFromUrl(url);
} else if (url.startsWith("bitcoin:")) {
    future = PaymentSession.createFromBitcoinUri(new BitcoinURI(url));
}

PaymentSession session = future.get();    // may throw
PaymentRequestException.
String memo = session.getMemo();
BigInteger amountWanted = session.getValue();
if (session.isExpired())
    showUserErrorMessage();

PaymentSession.PkiVerificationData identity = null;
try {
    identity = session.verifyPki();
} catch (Exception e) {
    log.error(e);
    // Don't show errors that occur during PKI verification to the user!
}
```

```
if (identity != null)
    showUserConfirmation(identity.domainName, identity.orgName);
else
    showUserConfirmation();

// a bit later when the user has confirmed the payment

Wallet.SendRequest req = session.getSendRequest();
wallet.completeTx(req);  // may throw InsufficientMoneyException
// No refund address specified, no user specified memo field.
ListenableFuture<PaymentSession.Ack> ack = session.sendPayment
(ImmutableList.of(req.tx), null, null);
Futures.addCallback(ack, new FutureCallback() {
    @Override public onSuccess(PaymentSession.Ack ack) {
        wallet.commitTx(req.tx);
        displayMessage(ack.getMemo());
    }
});
```

Resources

Bitcoinj Documentation:

https://code.google.com/p/bitcoinj/wiki/PaymentProtocol

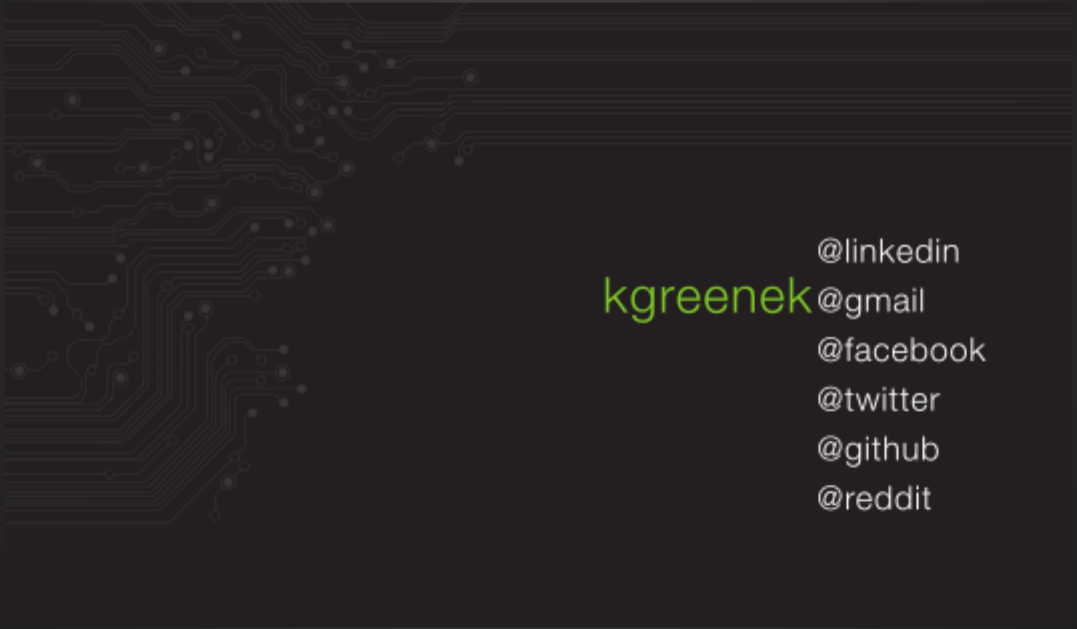
Handy-Dandy PaymentRequest Generator:

https://bitcoincore.org/~gavin/createpaymentrequest.php

Wallet-tool Sample Implementation:

https://github.com/bitcoinj/bitcoinj/tree/master/tools

Thank You!



kgreenek @linkedin

@gmail

@facebook

@twitter

@github

@reddit