Bitcoin Direct Payment

Optimal direct payment experience without loss of transaction privacy or integrity

Background

Our objective is to facilitate point of sale and peer-to-peer scenarios, which we here refer to as "direct payment". The most common existing technique is the optical scan of a Bitcoin URL encoded as a QR code. However QR codes can be tedious to scan, are susceptible to long range snooping, are unidirectional, and typically cannot carry enough data for more advanced payment scenarios. NFC can resolve the issues with one exception. User interaction requires device proximity twice during the process. We therefore propose a hybrid solution consisting of both NFC and Bluetooth technologies. For the sake of simplicity we use the general term Bluetooth to encompass related technologies such as Bluetooth Low Energy (BLE).

Roles

For the sake of clarity this document uses the names "customer" and "merchant" for the payer and receiver respectively and is not meant to exclude peer-to-peer payment scenarios.

References

Bitcointalk discussion of NFC/BT POS system

Example source using NFC and BT for simple payments

Basic Payment Scenario

The basic NFC payment scenario is a simple narrow cast or scan of a bitcoin URL (or in the form of a simple bitcoin address). Transfer can be sufficiently secured by proximity and there is no follow-on point-to-point communication necessary. The payment transaction is transferred directly over the customer's Internet connection.

By limiting the use of NFC to smaller amounts of data it may be possible to reduce cost. However it may be necessary to support longer URLs for NFC than for QR code scanning (see Payment Session Establishment).

The narrow cast and scan of a bitcoin URL/address via NFC should be compatible with existing implementations. Schildbach and Bridgewalker wallets are able to receive Bitcoin URIs via NFC.

Basic Direct Payment Scenario

The direct payment scenario is designed to preclude the need for a customer to have a network connection and, in the case where connectivity is available, to save the customer's battery. It also accelerates the receipt of the transaction by the merchant, which is beneficial in offline and/or low-risk scenarios. This is accomplished by the customer submitting the payment transaction via the merchant. This is most beneficial when the merchant is stationary with power and a network connection. This is a very common shopping scenario.

The basic scenario requires no session and should be compatible with existing implementations. Schildbach and Bridgewalker wallets are able to receive transactions via Bluetooth. These services advertise a Bluetooth service using the UUID **3357a7bb-762d-464a-8d9a-dca592d57d5b**. The Bluetooth MAC may be provided as an additional parameter in all Bitcoin URIs:

```
bitcoin:?bt=<BT MAC>
```

Clients that support this legacy convention can transmit a serialized Bitcoin transaction via Bluetooth using a simple format, as specified by the Schildbach wallet.

Note that the specification of a MAC address can prevent confusion in an area where there are multiple points of sale however this is not a security feature. A published MAC address is easily discovered and spoofed and is therefore insufficient for establishing a trusted connection. We support NFC URI publication of our MAC address and parsing such a published MAC address for compatibility purposes, but we should promote an improved security model as described in Payment Session Establishment. In any case no MAC address parameter is ever required.

Payment Session Establishment

A payment session is initiated via NFC based on the proximity of the customer to the merchant. The merchant narrow casts the payment URL and the customer scans it. The NFC URL includes a base58 encoded symmetric 128 bit session key parameter:

```
bitcoin:?s=<Session Key>
```

The proximity-based transfer of the key provides sufficient assurance that only the customer and merchant (via their trusted platforms) have access to the session. Proximity is fundamental to the integrity and privacy of the scenario. The session key may be used to establish a secure Bluetooth payment session. BTLE uses AES-CCM for session encryption so it should be possible to utilize the existing stack following the initial session handshake. However, reliance on closed-source implementations, or the baseband processor, is ill-advised. We therefore implement session encryption at the application layer.

This technique resolves significant vulnerabilities in Bluetooth key exchange. Proximity establishment is meant to prevent an interloper from hijacking either end of the payment session over the air. The secure establishment of the session also provides for privacy of the communication while still allowing the customer to physically back away from the NFC terminal while deciding whether to approve the transaction. The consumer can be shown whether the session is secured. The merchant has the option to require a secure session in construction of the URL.

Note that the session does not confirm the identity of the merchant. Identity is established via a signed payment request transferred over the session. So even if the session key is inadvertently exposed to an interloper there is no risk of payment misdirection if the merchant provides a signed payment and the customer verifies its presented alias. In the case of merchant signing certificate compromise, the merchant will eventually discover that transactions are being misdirected.

The Bluetooth payment session is established by way of a handshake. Starting at least at the time of a NFC narrow cast, the merchant broadcasts a Bluetooth service. The service accepts customer HELLO messages and responds with ACK messages, with both messages encrypted using the session key. Any connection attempt that does not decrypt to HELLO is considered invalid and rejected. All communication over a given connection is encrypted using the key established in the handshake. Session keys must be randomly generated and never reused in a subsequent session.

The strength of privacy is a function of each party's ability to maintain trustworthy platforms, and for the merchant to maintain sufficient distance between the NFC terminal and potential interlopers. The strength of payment integrity is a function of the ability of the merchant to avoid certificate compromise, and the practice of customers to visually verify merchant identity as indicated by the validated certificate.

The optional MAC address may be used by the customer to narrow the field of connection options.

```
bitcoin:?s=<Session Key>&bt=<BT MAC>
```

Any additional public information is superfluous as MAC addresses are presumed to be globally unique. However a per-customer differentiation may be derived from the session key and used to identify the customer publically to the service.

Advanced Payment Scenario

Since there is no existing implementation for the transfer of BIP70 payment requests and payment messages over Bluetooth we require that a private session be established. This requirement is further justified by the nature of the payment request model, in the significant potential for privacy and monetary loss.

Integration of BIP70 moves the payment address or addresses into session data. In simpler models the address is protected by proximity. However with BIP70 a MITM attack on a public session can easily steal money if the consumer is not diligent in checking the merchant's signature, or if the merchant is not signing. Furthermore, the exposure of a signed payment creates cryptographically strong tainting of transactions. Payment requests must be kept private to the parties in a transaction.

If a NFC scanned URL contains a payment request parameter, but no session key parameter:

```
bitcoin:?r=https://foo.co/7
```

then the payment request should be downloaded and the payment and the optional payment message sent directly over the customer's Internet connection (i.e. online payment request). If both parameters are present in the scanned URL:

```
bitcoin:?r=https://foo.co/7&s=12drXXUifSrRnXLGbXg8E
```

then the customer has the option of obtaining the payment request via an http/s or Bluetooth session. This provides for simpler NFC-only customer platforms.

If a payment address is present in the scanned URL:

```
bitcoin:1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa?r=https://foo.co/7&s=12drXXUifSrRnXLGbXg8E
```

then the payment request may be bypassed altogether. This provides for even simpler scenarios where the customer's platform does not support BIP70 or cannot create an http/s or Bluetooth payment session. If the URL is scanned via QR code then offline payment options should be ignored. This discourages publication of private data via dynamic QR codes, as optical presentation is subject to interception, and to limit the density of QR code presentations.

With a private session established the customer may request a serialized Payment Request from the merchant. If the customer subsequently decides to approve payment he/she may submit a BIP70 Payment Message (which encapsulates a transaction) to the merchant. This may save time in the transaction, improve customer privacy and conserve customer battery.

If not signed the Payment Request object is presumed to be valid, as the transport is presumed to implement error detection and correction, and the session is presumed to be private. The only guarantee of correctness requires BIP70 signature validation.

If the Payment Request object contains a Payment URL property:

```
payment_url=<URL>
```

then the customer has the option of submitting a Payment Message to the merchant in accordance with BIP70. If the Payment Request object contains a Session Payment property:

```
session_payment=true
```

then the customer has the option to submit a serialized Payment Message to the merchant over existing payment session. We do not allow the merchant to limit the customer's options in submitting a Payment Message.

The Payment Message is strongly-identified by the session. The customer should not include the Merchant Data property. The value is only reasonably useful for correlation, which is redundant. Even if signed the value is subject to a replay attack, so the merchant cannot trust the value therefore must ignore it if provided.

Submission of a Payment Message may or may not require a reconnection to the merchant's service, depending on how much time has passed. If, at the time of a reconnection attempt, the merchant has discarded the session key, the reconnection will not succeed.

The merchant is expected to submit the encapsulated transaction to the Bitcoin network. Alternatively, the customer may submit the transaction via an alternate channel and the merchant must monitor the network to receive it.

The response from a Payment Message is a BIP70 Payment Ack. The primary purpose of this message is to present a memo from the merchant to the customer. This is the final message in the session and results in automatic termination of the session, as each new payment session requires a new URL scan.

Offline Merchant

An offline merchant would not have any visibility into online transactions posted by a customer. However an offline merchant may receive and cache offline transactions as described above. If the offline merchant trusts that the transaction is sufficient he/she may opt to transfer merchandise despite the risk of loss, invalidity or double-spend of the coins transacted. For this reason it is generally advisable that the customer submit a Payment Message over the session.

Multiple Concurrent Payers

This section has been added on 2/7/2014, as an improvement of the Schildbach and Schroder proposal to include a Resource Name for customer differentiation. The original proposal is over-specified in that the hash of the Payment Request specified in the URL is sufficient. In our proposal there is no Resource Name or Payment Request hash in the URI, but a hash of the session key is equally suitable.

In an environment where multiple customers are paying it is beneficial to performance for the customer's initial connection to be established using a unique identifier provided by the merchant. Therefore the Bluetooth connection message is prepended with a hash of the session key prepended to the encrypted HELLO message. The merchant may use this value as a natural key for session key retrieval.

The hash function does not have to eliminate collisions, as the purpose is to narrow the merchant's search space, so we recommend a SHA1 hash truncated to 32 bits.

```
key=0x000102030405060708090a0b0c0d0e0f

BASE58(key)-> 12drXXUifSrRnXLGbXg8E

&s=12drXXUifSrRnXLGbXg8E&bt=

TRUNC32(SHA1(s)) -> 6e7da589
```

Generalization

This section has been added on 2/7/2014, with credit to the Schildbach and Schroder proposal entitled Extensible URI Parameter.

There will likely be evolution in session technologies and protocols. For example, there is a necessary distinction between support for BT and/or BLT. As such we should provide more flexibility than is available through the session key (s) and Bluetooth (bt) URL parameters above. This should be generalized with the idea of the BIP70 request parameter (r).

The Extensible URI Parameter proposal extends the BIP70 parameter (r), creating an ordered set of bitcoin URL parameter names rN, where the first member must not include the zero value (for simplicity in maintaining backward compatibility):

```
{ r, r1, r2, r3, … }
```

The proposal similarly overloads the Payment URL property for correlation, although this presents some complexity difficulty in terms of ensuring backward compatibility.

Incorporation

By incorporating the session key and the optional MAC address (in the case of Bluetooth at least) into a URL, the encoding allows for coupling the session key to the applicable session technology and generalizes nicely with the BIP70 URL parameter. Proper consideration should be to formalization and IANA registration of any URI schemes proposed for inclusion. These identifiers should be considered URLs in that their objective is to *locate* the resource, not just *identify* it.

A Bluetooth identifier, with the optional MAC address encoded as base58, yields:

```
bitcoin:?r=https://foo.co/42&r1=bt:12rAs9mM/12drXXUifSrRnXLGbXg8E
```

- Scheme: TLS
- Service Host: foo.co
- Resource Name: "42"
- Session Key: [TLS derived]

OR

- Scheme: Bluetooth
- Service UUID: 3357a7bb-762d-464a-8d9a-dca592d57d5b
- MAC Address: 00:10:60:B2:CE:6C
- Resource Name: "6e7da589"
- Session Key: 0x000102030405060708090a0b0c0d0e0f

Bluetooth and BLE combined, with a shared session key, no http/s option, and the MAC address specified only for the Bluetooth identifier yields:

```
bitcoin:?r=bt:12rAs9mM/12drXXUifSrRnXLGbXg8E&r1=ble:/12drXXUifSrRnXLGbXg8E
```

- Scheme: Bluetooth
- Service UUID: 3357a7bb-762d-464a-8d9a-dca592d57d5b
- MAC Address: 00:10:60:B2:CE:6C
- Resource Name: "6e7da589"
- Session Key: 0x000102030405060708090a0b0c0d0e0f

OR

- Scheme: Bluetooth Low Energy
- Service UUID: 3357a7bb-762d-464a-8d9a-000000000000
- MAC Address: -
- Resource Name: "6e7da589"
- Session Key: 0x000102030405060708090a0b0c0d0e0f

Because the session key is required it is included the hierarchy. The MAC address is optional, but logically higher than the session key, so a leading slash in the hierarchy indicates its absence. It may make sense to require the MAC address in order to facilitate discrimination in a crowded environment. Any evolution in the session key or MAC address can be accommodated in the hierarchy.

To retain the correlation feature, improve consistency and ensure backward compatibility, the Payment URL property could be handled in the same manner as the Payment Request property:

```
{ payment_url, payment_url1, payment_url2, payment_url3, … }
```

This has the added benefit of simplifying implementation, in that only the correlated property typically needs be read by the customer. The other members of the enumeration may be ignored (aside from the lower level protocol buffer parsing). It is also sufficient to use the parameter as a sentinel in the case where the scheme allows for payment in the session of the request.

However, because of the design of protocol buffers this approach lends itself to a limitation on the possible length of the enumeration. If it can be shown that there is no impact on backward compatibility, the Extensible URI Parameter technique would be comparable.

Base58 Encoding

Some considerations with respect to the choice of base58 encoding of MAC address and session key URL parameters:

Base64 is poorly specified, as there are more than a dozen variants.

All bitcoin libraries are likely to carry base58 but not probably base64. I recently added base64 to libbitcoin for support of message encryption/decryption, but that's a less common feature.

Base64's three byte alignment makes it less efficient than you might expect:

```
16: 001060b2ce6c
58: 12rAs9mM
64: ABBgss5s

16: 000102030405060708090a0b0c0d0e0f
58: 18DfbjXLth7AQ3TzT6mnqk
64: AAECAwQFBgcICQ0KCwwNDg8=
```

Base64 also has invalid URI characters, so the values will get URI encoded, making the actual values longer and less human-friendly. For example:

```
16: ffffffffffff
58: 3CUsUpv9t
64: %2F%2F%2F%2F%2F%2F%2F%2F
```

It's an extreme example, but base58 wins 9 to 24.