Secure Mobile Payments Without Network Connectivity

Abhijit Bera
Arun Samudrala
Pankaj Gupta
Dilip Padmanabhan
Hemanth Sambrani

See next page for additional authors

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation
Bera, Abhijit; Samudrala, Arun; Gupta, Pankaj; Padmanabhan, Dilip; Sambrani, Hemanth; Weng, Shuo; and Wong, Wallace, "Secure Mobile Payments Without Network Connectivity", Technical Disclosure Commons, (October 26, 2020)
https://www.tdcommons.org/dpubs_series/3700

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
Inventor(s)
Abhijit Bera, Arun Samudrala, Pankaj Gupta, Dilip Padmanabhan, Hemanth Sambrani, Shuo Weng, and Wallace Wong

This article is available at Technical Disclosure Commons: https://www.tdcommons.org/dpubs_series/3700
Secure Mobile Payments Without Network Connectivity

ABSTRACT

Mobile payments depend on the availability of internet connectivity, e.g., to enable a centralized service to authenticate a payment. This disclosure describes techniques to enable peer-to-peer mobile payments in the absence of a network. A user has an initial amount, referred to as the balance, that is transferred to their mobile device from a balance provider, e.g., a financial institution. The balance is digitally signed by both the user and the balance provider. To transact in the absence of a network, peer users perform a contactless payment as follows. The receiver of funds verifies the availability of funds by examining the prior, authenticated, transaction records of the sender. A transaction record including the transaction amount is created and made immutable and secure using cryptographic techniques. When either the sender or receiver regains network connectivity, the transaction is settled with the balance provider. Double-spend attempts by a malicious sender are forestalled by enabling secure maintenance of the true balance on a sender’s device (even in the absence of a network), and by enabling the receiver to settle with the sender’s balance provider on the basis of an authenticated transaction record.

KEYWORDS

- Mobile payment
- Peer-to-peer payment
- Networkless payment
- Double-spend attack
- Near-field Communication (NFC)
- Keypair
- Public-key cryptography
- Digital cash
- Decentralized payments
BACKGROUND

Digital payments depend on the availability of internet connectivity to enable a centralized service to authenticate a payment. The network dependency prevents mobile payments in the absence of the network, or even when there is a spike in network traffic. Even if sufficient data center capacity is provided to handle spikes in traffic, the bandwidth and coverage of the underlying mobile network remain a bottleneck. Also, the extra data center capacity, which requires investment, remains underutilized during non-peak times. Network connectivity remains a problem in many parts of the world. Even in relatively developed regions of the world, network connectivity can be a problem in densely-populated areas such as transit stations, shopping malls, tech parks, etc.

DESCRIPTION

This disclosure describes techniques to enable peer-to-peer mobile payments in the absence of a network, and on devices that may not have fully secure operating environments. When a mobile device of a user is connected to a network, the user requests an initial amount, referred to as the balance, to be transferred to their mobile device from a balance provider, e.g., a financial or other institution, using an acceptable financial instrument. With user permission, the mobile device is remotely tested for integrity, e.g., the absence of malicious hardware or software, by the balance provider using a mobile-device integrity verifier.

For example, the integrity of the device can be verified by having the device generating a keypair, and using the keypair to have the device attest itself with a root certificate issued by a trusted certificate authority. Upon successful remote verification of the integrity of the user device, the user is issued a user-identity bundle by an identity issue service (IIS) and a balance by the balance provider, digitally signed by both the user and the balance provider. For safety
reasons, the balance can have an expiry date, e.g., a date after which it cannot be used without reconfirmation by the balance provider.

The user-identity bundle enables peers to identify and authenticate each other during a transaction. The user-identity bundle can include data such as user name, user public key, receiver public key, identity expiry time, special permissions (if any), risk score, etc. The user-identity bundle is valid for a limited period of time; after its expiry, the credentials are refreshed by re-connecting to the IIS. Upon reconnection, the state of the user device and their previous transactions are validated before issuing a freshly-signed user-identity bundle, thus reducing the risk of abuse.

In the absence of a network, peer users bring their mobile devices near each other (or otherwise establish peer-to-peer communication between the devices) to effect a contactless payment (e.g., over Bluetooth or NFC), without either device authenticating itself over a network, as follows. The peer devices connect, authenticate each other, and set up a secure communication channel using corresponding user identities, which are previously issued and digitally signed by a trusted third party.

With appropriate permissions set up, the receiver device that is to receive funds verifies the availability of sufficient balance at the sender to cover the transaction by examining the prior, authenticated, transaction records of the sender, e.g., going back to the initial deposit made by the balance provider. For the current transaction, a new transaction record including the present transaction amount is created on both sender and receiver. The transaction record is signed with a transaction-specific keypair generated by the secure key storage of the sender’s device and attested to by the sender’s device (e.g. using a hardware-backed trusted execution environment of the device); made immutable by being signed by the private keys of both parties to the
transaction; and is appended to ledgers maintained at both sender and receiver device. When either the sender or receiver regains network connectivity, the transaction is settled with the balance provider. A sender can settle with all their counterparty receivers. A given receiver can settle with not only a given sender but with all the receiver-counterparties of the given sender who transacted with the given sender prior to the given receiver, without being privy to the other receivers’ transactions with the sender. The ability of any single receiver in the pool of receivers to settle other receivers’ transactions adds redundancy, e.g., even in an environment of generally poor network connectivity, transactions can be rapidly settled.

The use of transaction-specific keypairs ensures that any attempt by the sender to alter or delete a transaction from the ledger, e.g., to fraudulently claim a larger running balance or to cheat a recipient of a previous transaction, results in further transactions being disabled. Because the receiver also has a copy of the authenticated transaction record, attempts by the sender to deny a transaction fail, as the receiver can always settle with the sender’s balance provider upon the re-availability of a network connection. Double-spend attempts by the sender are forestalled by enabling a receiver to authenticate the true and current balance on a sender’s device (even in the absence of a network), and by enabling the receiver to settle with the sender’s balance provider on the basis of an authenticated transaction record. Cloning of the balance on another device is rendered ineffective by the user authentication and device integrity checks, and by binding the balance to the device.

Aside from peer-to-peer mobile payments, the techniques of this disclosure can also be used to enable unlocking of paid services such as transit gates, hotel rooms, self-driven, shared vehicles, etc.
With user permission, the techniques can leverage further features of mobile devices if available, such as:

- A hardware-backed secure key storage implemented in a trusted execution environment. Further, key material that resides on the hardware-backed secure key storage that cannot be extracted by pure software methods.
- Biometric authentication.
- A secure key storage that ensures that an application can access its own keys but not those of another application.
- A keypair attestation that provides a reliable indicator of genuineness of the secure key storage, its capabilities, the state of the bootloader, and the integrity of the application. A secure key storage that supports keypair attestation, which in turn provides a verifiable chain of trust from the bootloader to the application.
- Security mechanisms that ensure that compromising a mobile payments app and/or operating system (OS) at runtime is relatively non-trivial.

Key material that is non-extractable from the secure key storage via software methods ensures that secret keys cannot be stolen or tampered with. The secure key storage also ensures that the keys of one app/process cannot be used by another process in an uncompromised OS. Hardware security modules or secure elements provide an added benefit that hardware attacks to steal or tamper the secret keys are also non-trivial. Per the techniques, mobile devices that include the above features can perform purely offline transactions, e.g., where both sender and receiver are offline during the transaction.

Keypair attestation is a determining factor to establish trust between the two peers. Keypair attestation on hardware-backed secure key storage reliably indicates whether: the device
boodlecker is unlocked; the boot state is not verified; and/or the application has been modified. If the receiver finds that any of the above is true, the transaction with the sender can be declined or proceed to enforce online authentication of the transaction.

If the mobile device is compromised, e.g., by a reboot of the device to unlock the bootloader and subsequent installation of a rootkit, the previously issued balance gets destroyed. The user is required to obtain a fresh balance again, triggering a mobile-device integrity verification, which in turn indicates a compromised/unlocked device. The balance provider that issued the original balance can then deny issuing new balance to the user.

<table>
<thead>
<tr>
<th>Signature (key)</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private key (sk stands for secret key)</td>
<td>sk</td>
</tr>
<tr>
<td>Public key (pk stands for public key)</td>
<td>pk</td>
</tr>
<tr>
<td>The identity private key of a sender</td>
<td>sku</td>
</tr>
<tr>
<td>The identity private key of the 1st receiver</td>
<td>skr1</td>
</tr>
<tr>
<td>The public key of the nth transaction keypair</td>
<td>pk_n</td>
</tr>
<tr>
<td>A balance-provider (or identity-provider) I’s private key</td>
<td>sk_I</td>
</tr>
<tr>
<td>Signature operating on data T using private key sk</td>
<td>Ssk(T)</td>
</tr>
</tbody>
</table>

Table 1: Notation

Table 1 illustrates the notation used in this disclosure. The balance can be signed by the user and by the balance provider, which can be, e.g., on the RAM of a user device, and an immutable ledger of transactions, which can be, e.g., on the filesystem of the device. The balance provider issues a balance to the user, which remains immutable through its expiry. The issued balance, its expiry date, the user’s public key, along with other data such as maximum allowed transaction, transaction IDs, etc., constitute a balance bundle.
The balance bundle, B, is signed by the balance provider (and identity provider) using its private key, as indicated: $S_{skI}(B)$. The balance bundle is also signed by the private key of the user. The signature of the balance provider can be verified by a party that wishes to transact with the user, thereby verifying the authenticity of the issued balance. As explained before, the balance bundle is issued after checking the integrity of the user’s device via a mobile-device integrity verifier and by keypair attestation. This balance bundle is linked to an underlying form of payment (FOP) that the user has linked with their payment-app account. Some example FOPs are unified payment interface (UPI) mandate with multiple executions; prepaid wallet; postpaid credit; etc.

An immutable ledger of transactions is maintained by the sender and by the receiver. A transaction is made immutable by first signing the balance bundle with a per-transaction keypair generated by an app of the sender on the secure key storage of their device. This signature is called the transaction signature. When a balance is requested by the user, a zeroth keypair ($sk_0, pk_0$) is generated, which is used by the balance-provider to authenticate the user.

The first transaction keypair is ($sk_1, pk_1$), the second transaction keypair is ($sk_2, pk_2$), etc. The signing of the balance bundle with the first secret key is indicated by $S_{sk_1}(B + t_1)$, where $t_1$ is the timestamp of the transaction. The sender cannot directly access a transaction keypair because of the properties of the secure key storage. The sender cannot rollback to the previous transaction because the app overwrites the previous transaction’s keypair when the current transaction keypair is generated. The receiver can verify the transaction keypair via the secure key storage attestation and transaction signature. Both sender ($u$) and first receiver ($r_1$) then confirm the transaction signature by signing with their respective private keys, e.g., via the operations $S_{sku}(T_1)$ and $S_{skr_1}(T_1)$ for the first transaction, where $T_1$ is the transaction record,
e.g., the data structure that includes the transaction amount. This makes the transaction entry in the ledger immutable.

In a similar manner, the second transaction, in which the sender $u$ sends an amount encapsulated in transaction record $T_2$ to a receiver $r_2$ at a time $t_2$ is signed by a transaction signature $sk_2$, denoted $Ssk_2(B+t_2)$, signed by the sender’s signature $Ssku(T_1+T_2)$, and signed by the receiver’s signature $Sskr_2(T_1+T_2)$. For every signature made by a private key, the receiver obtains the corresponding public key via the secure channel, such that the receiver can verify the sender’s identity, balance in their account, the series of transactions conducted by the sender, etc.

**Threat models**

The described techniques can mitigate a variety of threats or attacks vectors, some of which are listed in Table 2.

<table>
<thead>
<tr>
<th>Attack vector</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender unlocks the bootloader, roots the OS (a common way of rooting), and uses a re-packaged unsigned app.</td>
<td>Assuming that the bootloader state is included with the keypair attestation (which is true on devices with a secure hardware-backed key storage), the receiver app will detect that the bootloader is unlocked and deny the transaction.</td>
</tr>
<tr>
<td>Sender attempts to repackage the app on an uncompromised OS.</td>
<td>An incorrect application signature will be reported to the secure key storage by the uncompromised OS; this can be detected by the receiver app.</td>
</tr>
<tr>
<td>Sender colludes with multiple receivers. Both sender and receiver run a re-packaged binary. The sender rolls back transactions and carries out multiple spends with the same balance bundle while the receivers skip the verification step. Then the receiver goes online to settle their transactions.</td>
<td>Since the balance provider has information on the balance amount issued to the sender it can reliably detect that the transaction amount received from that sender is higher than the balance issued. Also, when the receiver goes online to claim their money, the attestation from their secure key storage would be verified at the server, and</td>
</tr>
</tbody>
</table>
### Table 2: Various threats and their mitigations

<table>
<thead>
<tr>
<th>Attack vector</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>the unlocked-bootloader or incorrect app signature would be detected. The transaction can be flagged to be manually verified.</td>
<td>This threat is mitigated in a manner similar to the one where the payment application itself is re-packaged.</td>
</tr>
<tr>
<td>Malware installed on the sender’s phone gets access to keys generated by a payment application on the hardware-backed secure key storage.</td>
<td>The underlying cryptographic technique to establish the secure channel between the two parties prevents man-in-the-middle attacks. Additionally, verified identities for merchants with pictures are displayed on the phone of the user (sender of money).</td>
</tr>
<tr>
<td>Phishing attack by a man-in-the-middle pretending to be the receiver.</td>
<td>Making the transaction keys require biometric authentication will prevent the malicious person from performing transactions to some extent. An additional PIN entry within the payment app can prevent unauthorized transactions. Thus the stolen phone cannot be easily used for offline transactions.</td>
</tr>
<tr>
<td>User’s phone gets lost/stolen and gets into the hands of a malicious person.</td>
<td></td>
</tr>
</tbody>
</table>

### CONCLUSION

Mobile payments depend on the availability of a mobile network, e.g., to enable a centralized service to authenticate a payment. This disclosure describes techniques to enable peer-to-peer mobile payments in the absence of a network. A user has an initial amount, referred to as the balance, that is transferred to their mobile device from a balance provider, e.g., a financial institution. The balance is digitally signed by both the user and the balance provider. To transact in the absence of a network, peer users perform a contactless payment as follows. The receiver of funds verifies the availability of funds by examining the prior, authenticated,
transaction records of the sender. A transaction record including the transaction amount is created and made immutable by being signed by both parties to the transaction. It is additionally signed with a transaction-specific keypair generated by and attested to by the sender’s device, and is appended to ledgers maintained at both sender and receiver. When either the sender or receiver regains network connectivity, the transaction is settled with the balance provider. Double-spend attempts by the sender are forestalled by enabling verification of the true balance on a sender’s device (even in the absence of a network), and by enabling the receiver to settle with the sender’s balance provider on the basis of an authenticated transaction record.

REFERENCES

