


# A Spendable Cold Wallet from QR Video

Rafael Dowsley<sup>1</sup>, Mylène C. Q. Farias<sup>1</sup> <sup>a</sup>, Mario Larangeira<sup>3,\*</sup>,  
Anderson Nascimento<sup>4</sup> and Jot Virdee<sup>4</sup>

<sup>1</sup>Faculty of Information Technology, Monash University, Melbourne, Australia

<sup>2</sup>Department of Electrical Engineering, Universidade de Brasília, Brasília, Brazil

<sup>3</sup>Department of Mathematical and Computing Science, Tokyo Institute of Technology/IOHK, Tokyo, Japan

<sup>4</sup>School of Engineering and Technology, University of Washington, Tacoma, U.S.A.

Keywords: Digital Wallet, Cryptocurrency, Blockchain, QR Code.

Abstract: Hot/cold wallet refers to a widely used paradigm to enhance the security level of cryptocurrency applications that was proposed on Bitcoin Improvement Proposal 32. In a nutshell, after performing an initial setup in which the hot wallet receives partial information of the cold wallet in order to hierarchically generate (transaction receiving) addresses, the *cold wallet* stays offline, whereas the *hot wallet* is kept online. The initial transferred information enables the hot wallet to generate receiving addresses for both wallets, but it can only spend its own funds, *i.e.*, it cannot spend the funds in the cold wallet. This design conveniently mimics money storage in daily life: pocket money is kept in a less safe location, *e.g.*, a regular wallet, while life savings are kept in a more safe environment, *e.g.*, banking account. Note that the funds that land in offline addresses cannot be spent if the cold wallet is kept permanently offline. We propose a protocol and a technical solution to spend funds from a cold wallet without physically connecting it to *any network*. We designed and implemented a prototype for a system based on Optical Camera Communication (OCC) in a screen to camera setting, which can receive messages from a computer screen at the rate of over 150kB per second. Our system consists of a sequence of QR codes – a QR video. Our solution minimizes the possible attack vectors, including malware, by relying on optical communication yet providing a larger bandwidth than regular QR code based solutions.


## 1 INTRODUCTION

The design of distributed ledgers based on blockchain technologies is a hot research topic. Much of this interest is due to the potential of such technologies to enhance financial systems worldwide, allowing a more dynamic environment for the exchange of transactions, leveraging modern network systems to further automate banking systems. The central role of this technology is to register activities, *i.e.*, the transactions, in a distributed network of nodes.

**The Role of the Wallet.** The wallet is the actual component that generates the transactions and normally resides in mobile phones or desktop computers of the users. This component manages the cryptographic keys, (hierarchically) generates addresses, issues transactions and does the accounting of the mon-

etary value stored in the addresses of the ledger. It is a critical component, and security vulnerabilities in the wallet can have devastating consequences that cannot be solved in the consensus layer of blockchain solutions. It is no wonder that a line of development is the departure from pure software wallets to hardware-based ones in order to avoid, or mitigate, several common attack vectors such as malware and tampering. However, comparatively, wallets have received far less attention from the research community than the consensus protocol.

**Hybrid Approach: Hot/Cold Wallets.** Hardware wallets assure a higher level of security, however they are not as convenient as their software counterpart. In particular, some of its security emerges from being disconnected most of the time. It is accessed only when issuing a new transaction, *i.e.*, sign a message with the cryptographic secret key. A common strategy to obtain both convenience and security is to rely on two wallets at the same time: “hot” and “cold”

<sup>a</sup> <https://orcid.org/0000-0002-1957-9943>.

\*This work was supported by JSPS KAKENHI Grant Number JP21K11882.

wallets. While the former is permanently connected to the network, and therefore exposed to increased security risks, the latter is permanently disconnected (potentially a hardware wallet for enhanced security). In this setting, as long as the cold wallet does not issue a transaction, it is not required to be online, given that the hot wallet can incorporate information from the cold wallet to generate new (transaction receiving) addresses controlled without having to periodically receive secret information of the cold wallet.

This particular setting mimics the storage of fiat money in our daily life: pocket money in our regular wallet (the hot wallet) that is usually carried on daily activities, and the savings account (the cold wallet), which is kept in a much more secure environment. As simple as the analogy is, it also suits the analogous use cases for companies or exchanges which keep high values in more static internal accounts, leaving the more dynamic ones for the daily activities.

### **Cold Wallet Cannot Broadcast Transactions.**

Part of the cold wallet conservative approach for security is to be permanently kept offline. In terms of convenience of use, however, this is a limitation. The capability of securely receiving funds is not affected, given that the hot wallet has information regarding the hierarchical cold wallet key generation and can still generate fresh addresses for each single receiving transaction. And other parties accessing the ledger cannot link these transactions to a particular wallet. In other words, the cold wallet can receive an unlimited amount of funds securely using different addresses. However, if the cold wallet is kept permanently offline (for security reasons), then it cannot issue any transaction, as it cannot broadcast it in a convenient and secure way. This work proposes an alternative technique to circumvent this severe limitation.

## **1.1 Intuition of Our Solution: QR Video**

We propose a solution in which the cold wallet is kept *permanently* disconnected from the network, even when issuing a transaction. Our wallet is not exposed to malware or network risks since technically it is never online. This seems to be counter intuitive, however, in order to circumvent the early outlined restrictions, our protocol relies on a novel transmission channel between the cold and hot wallets. Whereas regular wallets connect with the ledger via a network or hardware (e.g., USB cable), we devise a high capacity optical channel between wallets.

Although our approach provides an alternative for isolating the cold wallet (thus obtaining a very strong security level) while keeping desirable functionalities,

we should remark that it requires specific hardware. It requires a camera and a display on both cold and hot wallets in order to capture/transmit the optical information. Other solutions available on the market rely on *static* code generation, i.e., regular QR codes, which impose a strict limit of 4000 characters (De, 2018; NGrave, 2020; Khan et al., 2019). Our solution yields a higher transmission rate since the hot wallet's display generates a *sequence* of QR codes (i.e., a QR video) which are captured by the cold wallet's camera yielding a higher transmission capacity. To the best of our knowledge, this work is the first to propose such an optical approach. This largely unexplored mean of communication prevents known malicious attempts to access the secret information contained in the device executing the cold wallet (as regular QR codes do, however our solution provides more transmission capacity). The cold wallet uses its display to show the transaction's information to the user. If the user confirms the transaction, it is signed internally by the cold wallet (and the state is updated). The cold wallet's display shows the QR video containing the signed transaction, which is read by the hot wallet's camera.

Our proposed channel based on optical communication has technical challenges of its own, which required several adaptations, including the development of the novel video approach. An urgent issue is the size of the transactions, given that static QR codes can contain a limited amount of data, and more modern systems may require larger addresses/transactions (more on that later and in the work of Karakostas et al. (Karakostas et al., 2020)). Additionally, the optical channel is very sensitive to the orientation of the devices, which causes a high ratio of errors in the transmission. A solution is to rely on error correcting codes, which enlarges the data to be sent and therefore renders the limited data capacity of QR codes even more challenging. In order to solve these problems, we have opted to introduce the concept of a QR video. The QR video consists of a sequence of QR codes organized as frames of a video. By stacking several QR code frames, we managed to transmit messages up to 100kB in less than a second. By adding a layer of error correction to our screen to camera communication system, we end up with a robust and reliable solution.

We consider the introduction of QR video and its application on hot/cold wallets the main technical contributions of this work.

## **1.2 Related Works**

The Bitcoin Improvement Proposal 32 (BIP32) (Wuille, 2017) introduced the Hierar-

chically Deterministic (HD) Wallet Standard. It departed from ideas on deterministic wallets by Maxwell et al. (Maxwell et al., 2014). Gutoski and Stebila (Gutoski and Stebila, 2015) pointed out a flaw in BIP32 in the light of deterministic wallets and proposed a fix. However, they used a very weak security model that does not consider the possibility of the adversary getting to know public keys and/or signatures corresponding to secret keys which were not compromised. In addition, their security guarantee is pretty weak (the adversary is only considered successful if he manages to recover completely the master secret key, instead of considering the standard notion of unforgeability). Fan et al. (Fan et al., 2019) also proposed a countermeasure to protect deterministic wallets. But their ad-hoc solution lacks a formal analysis or security proof. Courtois *et al.* (Courtois et al., 2014) presented a relevant review on the security status of wallet implementations, and highlighted how bad and correlated randomness can compromise their security. Other results regarding the effects of weak randomness were presented in (Bregel and Rossow, 2018; Breitner and Heninger, 2019).

More recently, Das et al. (Das et al., 2019) presented a long overdue formal treatment of (ECDSA-based) hot/cold wallet solutions that can be used with legacy cryptocurrencies such as Bitcoin or Ethereum. The formalization in (Das et al., 2019) differs from the BIP32 specification in the fact that the wallets share a state. They proved that their solution satisfies unforgeability and unlinkability properties: unforgeability guarantees that as long as the cold wallet is not compromised, it is not possible to forge signatures to authenticate new transactions as being signed with the secret key corresponding to the cold wallet; while unlinkability guarantees that public keys derived from the same master public key are computationally indistinguishable from freshly generated public keys, and thus a recipient can receive multiple transactions without allowing other users to detect that the transactions belong to the same recipient.

Special hardware is an alternative for enhancing the security of wallets. Despite have been commercially available for some time, only recently this type of wallet has received a thorough formal analysis by Arapinis et al. (Arapinis et al., 2019). In particular, the authors considered attacks against the hardware of the wallet in the UC framework. Marcedone et al. (Marcedone et al., 2019) integrated two-factor authentication into hardware wallet solutions. In the realm of Proof-of-Stake (PoS), wallets potentially have more actions at its disposal than only issuing payment transactions (*e.g.*, *stake delegation*). To the best of our knowledge, only the work of Karakostas et

al. (Karakostas et al., 2020) proposes a thorough security analysis in the UC Framework of PoS-based wallets. As emphasized in (Karakostas et al., 2020), PoS addresses are larger than regular BIP32 ones, because they encode more information on staking delegation, staking pools and etc, therefore our QR video based solution, with higher transmission capacity, seem to be a better fit than QR code based ones.

## 2 STATEFUL HOT/COLD WALLET SCHEME

For completeness, we review (Das et al., 2019) which underpins our protocol to be presented in Section 3.

**Definition 1** (Stateful Hot/Cold Wallet Scheme (Das et al., 2019)). For a security parameter  $\lambda$ , a Stateful Hot/Cold Wallet Scheme is defined by the algorithms (MGen, SKDer, PKDer, WSign, WVer) such that:

- The probabilistic master key generation algorithm MGen takes as input the security parameter  $1^\lambda$  and outputs an initial state  $st_0$  that is given to both hot and cold wallets, a master public key  $mpk$  that is given to the hot wallet, and a master secret key  $msk$  that is given to the cold wallet;
- The deterministic secret key derivation algorithm SKDer is run by the cold wallet to derive a session secret key. It takes as input the master secret key  $msk$ , the state  $st$ , identifier  $ID$ , and outputs a session secret key  $sk_{ID}$  and the new state  $st'$ ;
- The deterministic public key derivation algorithm PKDer is run by the hot wallet to derive a session public key. It takes as input the master public key  $mpk$ , the state  $st$  and an identifier  $ID$ , and outputs a session public key  $pk_{ID}$  and the new state  $st'$ ;
- The probabilistic signing algorithm WSign is executed by the cold wallet to sign messages. It takes as input a message  $m$  and a session secret key  $sk$  and outputs a signature  $\sigma$ ;
- The deterministic verification algorithm WVer is executed by any party that wants to verify the validity of a signature. It takes as input a session public key  $pk$ , a message  $m$  and a signature  $\sigma$ , and outputs 0 (reject) or 1 (accept).

**Remark.** The identifier  $ID$  is used to construct a new public key  $pk_{ID}$  for an arbitrary choice of  $ID$  by allowing a common secret string  $ch$  which is called the “chaincode” between the wallets. As argued in (Das et al., 2019; Maxwell et al., 2014), for the ECDSA signature scheme, in order to derive a new key  $pk_{ID}$  while preserving the unlinkability and unforgeability security properties (more about the security games in Section 3.3), it is necessary to com-

pute  $w \leftarrow H(\text{ID}, ch)$ ,  $\text{pk}_{\text{ID}} \leftarrow \text{mpk} + w \cdot G$  and  $\text{sk}_{\text{ID}} \leftarrow \text{msk} + w$ , for a hash function  $H$  and ECDSA key pair  $\text{msk} = x$  and  $\text{mpk} = x \cdot G$ . For readability we omit the value  $ch$  in the description of our protocol.

**Correctness of a Stateful Hot/Cold Wallet Scheme.**

The correctness requirement guarantees that if the cold/hot wallets derive session key pairs on the same set of identifiers  $\text{ID}_1, \dots, \text{ID}_n$  and in the same order (departing from the initial state  $\text{st}_0$ ), then any signature created by the cold wallet using one of the session secret keys  $\text{sk}_{\text{ID}_i}$  should be accepted when the verification algorithm is executed with the corresponding session public key  $\text{pk}_{\text{ID}_i}$ . In other words, all the derived session secret and public keys should match.

**Remark.** Later in our protocol we assume the existence of a deterministic function  $I : \mathbb{N} \rightarrow \{0, 1\}^*$  to generate unique identifiers  $\text{ID}_1, \dots, \text{ID}_n$ , to identify the key sessions between the wallets. Concretely, given the sequential index  $i \in \mathbb{N}$ , both wallets can be kept synchronized by the correct generation of  $\text{ID}_i$  values, *i.e.*, executing  $I(i) = \text{ID}_i$ . For an arbitrary  $\text{ID}^*$  value, the cold wallet can check the equality  $I(i) = \text{ID}_i = \text{ID}^*$  for sequential values  $i$  and confirm the correct index for the session key  $\text{ID}^*$ . This confirmation procedure allow the pair of wallets to be kept synchronized regarding the session keys.

**Security of the Stateful Hot/Cold Wallet Scheme.**

As described by Das et al. (Das et al., 2019), the Stateful Hot/Cold Wallet Scheme should satisfy two security properties: *Unlinkability* and *Unforgeability*. Intuitively, the former protects the privacy of the transaction receiver and captures the guarantee that it should be infeasible to link transactions sending funds to different session public keys that were derived from the same master public key. It is required that an adversary that is given the master public key cannot distinguish between session public keys derived from that master public key, and session public keys that are generated from a fresh (*i.e.*, independently and randomly chosen) master public key. As highlighted in (Das et al., 2019), such security property cannot be achieved for keys which the adversary knows the state used to derive them (the adversary can learn such state if there is a breach in the hot wallet, for example).

The Unforgeability property captures the feature that once funds are transferred to the cold wallet, they remain secure if: (1) the hot wallet is breached; and (2) the adversary observes transactions signed by the cold wallet. Even such adversary cannot generate new valid transactions spending funds from the cold wallet. Section 3.3 reviews the security games.

In this work, we focus on the orthogonal problem of how to enable the cold wallet to generate spending transactions without being connected to the network, in a way that breaches of the cold wallet are avoided while keeping it fully functional. We assume, as a departure point, a Stateful Hot/Cold Wallet Scheme that is correct, unlinkable and unforgeable.

### 3 THE QR VIDEO BASED WALLETS

We now detail our proposed solution. Since our approach adds special features on both the cold  $W_{\text{cold}}$  and hot  $W_{\text{hot}}$  wallets, *i.e.*, the camera and display based communication, it is convenient to review the traditional hot/cold setting with an *unspendable* cold wallet (which contrasts with our *spendable* cold wallet approach) before we fully describe the protocol. In a traditional hot/cold wallet setting the sensitive information is handled by  $W_{\text{cold}}$ , while  $W_{\text{hot}}$  interacts with the ledger and provides the required information for generating a transaction (*i.e.*, destination address, amount, etc). Once the initial setup between wallets is finished, they do not interact anymore. That setting is convenient from the security point of view as the cold wallet cannot broadcast transactions to the network. In contrast, our cold wallet issues transactions and interacts with the hot wallet through the optical/QR video channel. This change in design has two major consequences we need to address: (1) the introduction of a communication protocol between both wallets; and (2) the guarantee that both wallets have the necessary hardware to execute the protocol.

#### 3.1 Description of the Setting

**Communication.** In our setting, after the initial setup phase, the cold wallet  $W_{\text{cold}}$  still interacts with the hot wallet  $W_{\text{hot}}$ , while keeping the secret-key. The communication protocol is used for exchanging the transaction attributes (from  $W_{\text{hot}}$  to  $W_{\text{cold}}$ ) and for receiving the signed transaction (from  $W_{\text{cold}}$  to  $W_{\text{hot}}$ ). We remark that while  $W_{\text{cold}}$  is only accessible via an optical communication channel,  $W_{\text{hot}}$  has a regular network connection in addition to the optical communication channel with  $W_{\text{cold}}$ .

**Hardware Requirements.** The cold wallet  $W_{\text{cold}}$  communicates with  $W_{\text{hot}}$  using a bi-directional optical channel (a screen and a camera), and this is the only mean of communication available for  $W_{\text{cold}}$  (there is no connection using USB cables, as common in hardware wallets, NFC, WiFi, Bluetooth or other com-



munication channels). The consequence is that both devices have to be equipped with camera and display. This optical channel allows connectivity while severely limiting the actions of a potentially corrupted  $W_{\text{hot}}$  wallet. Furthermore, the display of  $W_{\text{cold}}$  allows an independent verification of the received information by the user. A regular hardware-based wallet could in principle allow such interactivity using a physical connection, however with disadvantages. The most immediate drawback is that physical connections are easier to explore in attacks. Moreover, a cold wallet without a screen does not allow the independent verifiability of the exchanged information, *i.e.*, attributed and signed transactions.

### 3.2 The Communication Protocol

The starting point of our protocol is Definition 1; however, our construction puts forth a protocol for the exchange of messages between  $W_{\text{cold}}$  and  $W_{\text{hot}}$  aided by the optical channel in order to allow  $W_{\text{cold}}$  to generate signed transactions. The cold  $W_{\text{cold}}$  and hot  $W_{\text{hot}}$  wallets are described below. In the description of the protocol, each interface also specifies the equipment activated on each phase. For simplicity, we left out of the protocol description any sort of authentication between the two wallets. However we emphasize that a key/credential exchange protocol between the wallets can easily be added to the **Initialization** phase.

#### Cold Wallet $W_{\text{cold}}$

The cold wallet keeps variables for the current state  $\text{CurrentST}$  and current identifier  $\text{CurrentID}$ , which are initialized as  $\perp$ . It also keeps two initially empty lists  $\text{IDs}$  and  $\text{STs}$ , for identifiers and states.

**Initialization:** The user decides to perform the initial setup. The probabilistic master key generation algorithm  $(\text{mpk}, \text{msk}, \text{st}_0) \leftarrow \text{MGen}(1^\lambda)$  is executed to generate the master public key  $\text{mpk}$ , the master secret key  $\text{msk}$  and the initial state  $\text{st}_0$ . It then sets  $\text{CurrentST} \leftarrow \text{st}_0$ , adds it to  $\text{STs}$  and outputs in the display the message that the initialization is complete without showing any internal information.

**Display Initialization (QR Video):** The user opens the optical channel to transmit the master public key  $\text{mpk}$  and the initial state  $\text{st}_0$  to the hot wallet. The QR video encoding this information is shown in the display.

**Load Transaction Data (Camera):** The user chooses to receive the information regarding a transaction, then it receives the QR video which encodes  $(\text{tx}, \text{ID}, \text{st})$ . If  $\text{ID} \in \text{IDs}$  and  $\text{st} \in \text{STs}$ , then set  $\text{CandidateTX} \leftarrow \text{tx}$ ,  $\text{CandidateID} \leftarrow \text{ID}$  and  $\text{CandidateST} \leftarrow \text{st}$ . Otherwise

- set only  $\text{CandidateTX} \leftarrow \text{tx}$ , and interactively execute  $(\text{sk}, \text{st}) \leftarrow \text{SKDer}(\text{msk}, \text{CurrentST}, \text{CurrentID})$  and the identifier generation function  $I$ , until  $I(i) \rightarrow \text{ID}$  for some  $i$ ;
- in the process, update accordingly the lists  $\text{IDs}$  and  $\text{STs}$ , and the values  $\text{CurrentST}$  and  $\text{CurrentID}$ ;
- set  $\text{CandidateID} \leftarrow \text{ID}$ , and  $\text{CandidateST} \leftarrow \text{st}$ .

**Issue Transaction:** The user verifies the transaction information  $\text{tx}$  that is shown in the display and decides if it is approved or not. If it is, execute  $(\text{sk}_{\text{CandidateID}}, \text{st}) \leftarrow \text{SKDer}(\text{msk}, \text{CandidateST}, \text{CandidateID})$ , and signs the transaction  $\sigma \leftarrow \text{WSign}(\text{tx}, \text{sk}_{\text{CandidateID}})$ .

**Deliver Transaction (QR Video):** When the user chooses to release the signed transaction, then the wallet shows in the display the QR video encoding  $(\text{tx}, \sigma)$ .

#### Hot Wallet $W_{\text{hot}}$

The hot wallet keeps variables for the current state  $\text{CurrentST}$  and current identifier  $\text{CurrentID}$ , which are initialized as  $\perp$ . It also keeps two initially empty lists  $\text{IDs}$  and  $\text{STs}$ , for identifiers and states.

**Initialization (Camera):** The user chooses to initialize the wallet and starts the camera. The wallet then receives  $(\text{mpk}, \text{st}_0)$  from  $W_{\text{cold}}$  via QR video, and sets  $\text{CurrentST} \leftarrow \text{st}_0$ .

**Generate Session Public Key:** The wallet runs  $(\text{pk}_{\text{CurrentID}}, \text{st}) \leftarrow \text{PKDer}(\text{mpk}, \text{CurrentST}, \text{CurrentID})$ , the deterministic public key derivation algorithm, to obtain the session public key  $\text{pk}_{\text{CurrentID}}$ . Then, with  $I$  and the newly generated state  $\text{st}$ , it updates accordingly  $\text{CurrentST}$  and  $\text{CurrentID}$ , and the lists  $\text{IDs}$  and  $\text{STs}$ .

**Submit Transaction (Display QR Video):** Given a transaction  $\text{tx}$ ,  $\text{ID} \in \text{IDs}$  and

$st \in STs$ , generate a QR video which encodes  $(tx, ID, st)$  and make the transaction  $tx$  available on the display.

**Receive Signed Transaction (Camera):** The user chooses to receive the signed transaction and starts the camera. The camera is expected to read the QR video, which encodes  $(tx, \sigma)$ .

**Remark.** The reader should notice that in the *Generation Session Public Key* phase, we purposely did not highlight the interface to be used by the hot wallet. The public key information is to be handled as the receiving (often fresh) addresses for funds. It is not to be used with  $W_{cold}$ , therefore any regular mean of communication can be used.

### 3.3 Security Games of the Protocol

By relying on the model of Definition 1, we can also take advantage of the security analysis presented in (Das et al., 2019). Namely, our construction inherits the security guarantees of that work, *i.e.*, unlinkability and unforgeability. We observe that in (Das et al., 2019) the authors remark that the games capture the case where the adversary receives signatures, which is a condition not natural, given that the cold wallet is usually kept offline. On the other hand, this condition is concretely the case of our framework, since we assume the QR video channel between the cold and hot wallets, allowing the adversary to query for signatures/transactions of its choice.

Regarding the proposed security games, intuitively, the Unlinkability Game **Unl** presents the adversary with a signature from a wallet out of two possible, and it is requested to successively decide it, as in a standard indistinguishability game. While the Unforgeability Game **WUnf** resembles the regular unforgeability property for signature schemes.

For completeness, we review the games from (Das et al., 2019) in Tables 1 and 2. For the full security discussion of the framework we refer the reader to (Das et al., 2019).

## 4 IMPLEMENTATION: QR VIDEO

We now describe our solution in more detail. Firstly by describing the specification of the prototype, then we present pictures of the prototype in operation.

### 4.1 The Overall Setting

We divide the section in four items:

- **The Reader:** Description of the reader platform and the limitations of it;
- **The QR Codes:** Here we describe the error correction codes, and the limits in the amount of data;
- **The QR Video:** The novel approach to increase the data exchange ratio with video;
- **The Hardware Setting:** The hardware setting where the experiments were conducted.

**The Reader.** We have implemented our QR video reader by using the *Google's Barcode API*. For the sake of prototyping our solution, we have used an Android cell phone. However, in practice, this device should be a dedicated one, working solely as a wallet and not being connected to the internet at all. We present a sequence of QR codes to the wallet in a loop. Once a QR code is detected, it is then added to a hash set. The hash set is used to prevent duplicate detections. The API stops after it has reached 100,000 bytes which is thirty-five frames assuming each QR code holds 2880 bytes. This idea is expanded later in the QR video description. Once 100,000 bytes is reached, the data in the hash set is converted into a string and is then processed. The limit of 100kB was chosen based on the fact that it is enough for reliably transmitting the usual transactions appearing in current blockchain platforms, but it could be trivially modified to a larger number at the price of increasing the duration of the optical transmission.

**The QR Codes.** The QR video holds QR codes generated with a 7% error correction rate (Reed-Solomon code) and 1665 by 1665 pixels. In our tests, such error correction rate was sufficient to ensure a reliable communication between the wallets. Assuming a generic case where words and spaces are used in the QR code, Table 3 shows the largest amounts of data each level can hold. Also, each QR code is generated at the largest possible size,  $177 \times 177$  modules.

**The QR Video.** A QR video is a video holding multiple QR codes. Each frame holds a unique QR code. The length of video varies depending on how much data needs to be transferred. In our application, 100,000 bytes are needed, and split up into thirty-five 2880-byte QR codes. The codes are put together to create a video. The video lasts about 600 milliseconds. The video is rendered at 60 frames/second due to display constraints. Our transmission rate can be computed as follows:  $2880B \times 60 = 172.8kB/s$

Table 1: The adversary has access to two oracles, PK and WalSign, which respectively provides verification keys for a given session, and signatures for a given session.

<p><b>Main WUnf</b>  <math>(st, msk, mpk) \leftarrow MGen(\lambda)</math>  <math>(m^*, \sigma^*, ID^*) \leftarrow A^{PK, WalSign}(mpk, st)</math>  If <math>Keys[ID^*] = \perp</math>  Return 0  <math>(pk_{ID^*}, sk_{ID^*}) \leftarrow Keys[ID^*]</math>  If <math>m^* \in Sigs[ID^*]</math>  Return 0  If <math>WVer(pk_{ID^*}, \sigma^*, m^*) = 0</math>  Return 0  Return 1</p>	<p><b>Oracle WalSign</b>(m, ID)  If <math>Keys[ID] = \perp</math>: Return <math>\perp</math>  <math>(pk_{ID}, sk_{ID}) \leftarrow Keys[ID]</math>  <math>\sigma \leftarrow WSign(sk_{ID}, m)</math>  <math>Sigs[ID] \leftarrow Sigs[ID] \cup \{m\}</math>  Return <math>\sigma</math></p> <p><b>Oracle PK</b>(ID)//Once per ID  <math>st' \leftarrow st</math>  <math>(sk_{ID}, st) \leftarrow SKDer(msk, ID, st')</math>  <math>(pk_{ID}, st) \leftarrow PKDer(mp_k, ID, st')</math>  <math>Keys[ID] \leftarrow (pk_{ID}, sk_{ID})</math>  <math>Sigs[ID] \leftarrow \emptyset</math>  Return <math>pk_{ID}</math></p>
--	---

Table 2: The adversary has access to the PK, WalSign, Chall, and getSt oracles which provides, respectively, newly generated verification keys for a given session, signatures for a given session, the challenge verification key and the status of the hot wallet.

<p><b>Main Unl</b>  <math>(st, msk, mpk) \leftarrow MGen(\lambda)</math>  <math>b \xleftarrow{\\$} \{0, 1\}</math>, <math>Orc \leftarrow \{PK, WalSign, Chall, getSt\}</math>  <math>b' \xleftarrow{\\$} A^{Orc}(mpk)</math>  If <math>b' = b</math>: Return 1  Return 0</p> <p><b>Oracle WalSign</b>(m, ID)  If <math>Keys[ID] = \perp</math>: Return <math>\perp</math>  <math>(pk_{ID}, sk_{ID}) \leftarrow Keys[ID]</math>  <math>\sigma \leftarrow WSign(m, sk_{ID})</math>  Return <math>\sigma</math></p> <p><b>Oracle PK</b>(ID)  If <math>Keys[ID] \neq \perp</math>  <math>(pk_{ID}, sk_{ID}) \leftarrow Keys[ID]</math>  Else  <math>st' \leftarrow st</math>  <math>(sk_{ID}, st) \leftarrow SKDer(msk, ID, st')</math>  <math>(pk_{ID}, st) \leftarrow PKDer(msk, ID, st')</math>  <math>Keys[ID] \leftarrow (pk_{ID}, sk_{ID})</math>  Return <math>pk_{ID}</math></p>	<p><b>Oracle Chall</b>(ID)//One time  If StateQuery: Return <math>\perp</math>  If <math>Keys[ID] \neq \perp</math>: Return <math>\perp</math>  // Generate real key  <math>st' \leftarrow st</math>  <math>(sk_{ID}^0, st) \leftarrow SKDer(msk, ID, st')</math>  <math>(pk_{ID}^0, st) \leftarrow PKDer(msk, ID, st')</math>  //Generate random key  <math>(\widehat{st}, \widehat{msk}, \widehat{mpk}) \leftarrow MGen(\lambda)</math>  <math>(\widehat{sk}_{ID}^1, \cdot) \leftarrow SKDer(\widehat{mpk}, ID, \widehat{st})</math>  <math>(\widehat{pk}_{ID}^1, \cdot) \leftarrow PKDer(\widehat{mpk}, ID, \widehat{st})</math>  <math>Keys[ID] \leftarrow (pk_{ID}^b, sk_{ID}^b)</math>  Return <math>pk_{ID}^b</math></p> <p><b>Oracle getSt</b>  StateQuery <math>\leftarrow</math> True  Return st</p>
--	--

Table 3: Size of QR Codes in a Generic Case.

Error Correction Level	Amount of Data	Word Count
7%	2880 bytes	403
15%	2331 bytes	315
25%	1663 bytes	230
30%	1273 bytes	171

Note that it takes a couple of seconds for the user to align the camera with video and then hold a stable position for the reader to recognize the QR codes. We also remark that the QR video is looped for the user to receive all the necessary data.

**The Hardware Setting.** For this prototype, two devices were used. An android device to receive information and a laptop screen capable of clearly displaying the QR video (sender). The Android uses its camera at 1920 by 1080 resolution and 60 frames per second to scan the QR video.

## 4.2 Images of the Prototype

Figures 1 and 2 illustrate our protocol running the QR video in a mobile phone and computer setting equipped with cameras and display.

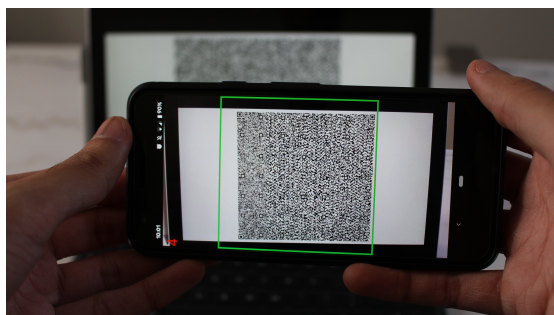


Figure 1: Point of view use of the QR-Video scanner.

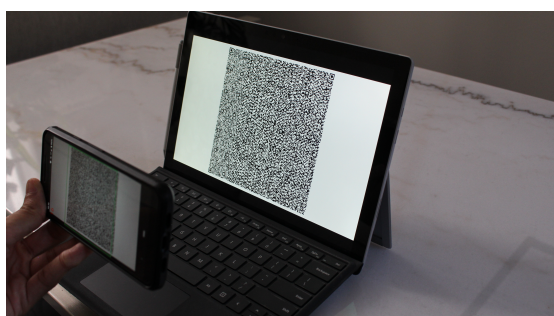


Figure 2: Side view use of the QR-Video scanner.

## 5 CONCLUSION

We proposed a new cold wallet communication setting: an optical (display to camera) channel. In our solution, a cold wallet equipped with a display and a camera can receive transactions through camera, display the transaction to the user for confirmation, sign the confirmed transactions, and display the signature on its display which is, in turn, acquired by a hot wallet's camera. Due to the size of the data transmission from the hot to the cold wallet, we proposed a novel encoding information encoding scheme - a QR video, *i.e.*, sequence of frames containing individual QR codes. Our solution achieves transmission rates over 150 kB per second. Thus, our cold wallet can authorize transactions not connected to the internet nor wired to another device (*e.g.*, through a USB port).

## REFERENCES

Arapinis, M., Gkaniatsou, A., Karakostas, D., and Kiayias, A. (2019). A formal treatment of hardware wallets. In Goldberg, I. and Moore, T., editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 426–445, Frigate Bay, St. Kitts and Nevis. Springer, Heidelberg, Germany.

Breitner, J. and Heninger, N. (2019). Biased nonce sense: Lattice attacks against weak ECDSA signatures in

cryptocurrencies. In Goldberg, I. and Moore, T., editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 3–20, Frigate Bay, St. Kitts and Nevis. Springer, Heidelberg, Germany.

Brengel, M. and Rossow, C. (2018). Identifying key leakage of bitcoin users. In Bailey, M., Holz, T., Stamatogiannakis, M., and Ioannidis, S., editors, *Research in Attacks, Intrusions, and Defenses*, pages 623–643, Cham. Springer International Publishing.

Courtois, N. T., Emirdag, P., and Valsorda, F. (2014). Private key recovery combination attacks: On extreme fragility of popular bitcoin key management, wallet and cold storage solutions in presence of poor RNG events. *Cryptology ePrint Archive*, Report 2014/848. <http://eprint.iacr.org/2014/848>.

Das, P., Faust, S., and Loss, J. (2019). A formal treatment of deterministic wallets. In Cavallaro, L., Kinder, J., Wang, X., and Katz, J., editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 651–668. ACM Press.

De, N. (2018). Crypto Wallet to Replace Private Keys With Encrypted QR Codes. <https://www.coindesk.com/markets/2018/07/27/crypto-wallet-to-replace-private-keys-with-encrypted-qr-codes/>. [Online; accessed August-2021].

Fan, C.-I., Tseng, Y.-F., Su, H.-P., Hsu, R.-H., and Kikuchi, H. (2019). Secure hierarchical bitcoin wallet scheme against privilege escalation attacks. *International Journal of Information Security*, pages 1–11.

Gutoski, G. and Stebila, D. (2015). Hierarchical deterministic bitcoin wallets that tolerate key leakage. In Böhme, R. and Okamoto, T., editors, *FC 2015: 19th International Conference on Financial Cryptography and Data Security*, volume 8975 of *Lecture Notes in Computer Science*, pages 497–504, San Juan, Puerto Rico. Springer, Heidelberg, Germany.

Karakostas, D., Kiayias, A., and Larangeira, M. (2020). Account management in proof of stake ledgers. In Galdi, C. and Kolesnikov, V., editors, *Security and Cryptography for Networks*, pages 3–23, Cham. Springer International Publishing.

Khan, A. G., Zahid, A. H., Hussain, M., and Riaz, U. (2019). Security of cryptocurrency using hardware wallet and qr code. In *2019 International Conference on Innovative Computing (ICIC)*, pages 1–10. IEEE.

Marcedone, A., Pass, R., and shelat, a. (2019). Minimizing trust in hardware wallets with two factor signatures. In Goldberg, I. and Moore, T., editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 407–425, Frigate Bay, St. Kitts and Nevis. Springer, Heidelberg, Germany.

Maxwell, G. et al. (2014). Deterministic wallets.

NGrave (2020). NGRAVE uses QR codes to keep its hardware wallet 100% offline. <https://medium.com/ngrave/ngrave-uses-qr-codes-to-keep-its-hardware-wallet-100-offline-f1e18be317a2>. [Online; accessed August-2021].

Wuille, P. (2017). Hierarchical Deterministic Wallets. <https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>. [Online; accessed July-2019].