Private Key Recovery Combination Attacks:
On Extreme Fragility of Popular Bitcoin Key Management, Wallet and Cold Storage Solutions in Presence of Poor RNG Events

Nicolas T. Courtois[1] Pinar Emirdag[2] Filippo Valsorda[3]

[1] University College London, UK
[2] Independent market structure professional, London, UK
[3] CloudFlare, London, UK

Abstract. In this paper we study the question of key management and practical operational security in bitcoin digital currency storage systems. We study the security two most used bitcoin HD Wallet key management solutions (e.g. in BIP032 and in earlier systems). These systems have extensive audit capabilities but this property comes at a very high price. They are excessively fragile. One small security incident in a remote corner of the system and everything collapses, all private keys can be recovered and ALL bitcoins within the remit of the system can be stolen. Privilege escalation attacks on HD Wallet solutions are not new. In this paper we take it much further. We propose new more advanced **combination attacks** in which the security of keys hold in cold storage can be compromised without executing any software exploit on the cold system, but through security incidents at operation such as **bad random number or related random events**.

In our new attacks all bitcoins over whole large security domains can be stolen by people who have the auditor keys which are typically stored in hot systems connected to the Internet and can be stolen easily. Our combination attacks allow to recover private keys which none of the earlier attacks in isolation could hope to recover. Classical bad random attacks typically concern only very few bitcoin accounts, and only some very lucky holders of bitcoins can actually steal other people's bitcoins. In this paper we go beyond identical random attacks and show several attacks which also work with related random events, which events are more probable and yet less likely to be detected before it is too late. We also present several attacks which work across distinct security domains which share no common setup, code or keys. Yet in certain circumstances **all the bitcoins in each domain can be stolen**. All our attacks are practical and realistic given the numerous relevant events have already happened in the bitcoin blockchain hundreds of times, some as recently as September 2014.

It is not clear if this problem can be repaired, i.e. if there exists a key management solution with similar audit capabilities as BIP032 which would be immune against this sort of advanced combination attacks.

Key Words: applied cryptography, bitcoin, key management, security management, audit capability, digital signatures, ECDSA, HD Wallets, BIP032, privilege escalation attacks, bad RNG, RFC6979.

1 Introduction

Bitcoin has been in existence for more nearly 6 years [15]. It is a digital currency, payment and final clearing/settlement system and technology, a distributed property register and digital notary service, all in one. Current bitcoin suffers from a number of obvious and well known technical problems: slow transactions acceptance [8], large storage at all full network nodes, poor anonymity [18, 13], high volatility, cyber attacks, to name just a few. In theory many different digital currency systems could exist [3]. In practice no real alternatives to bitcoin exist: all major digital currencies are essentially clones of bitcoin with small variations. They massively reproduce bitcoin, massively re-use open source code of bitcoin, and inherit all the security problems of bitcoin such as studied in this paper.
Bitcoins cultivates a certain type of "cryptographer's dream" in which participants do not trust each other, yet are able to somewhat function through pure magic of modern cryptography [3, 9] rather than through trusting some financial institutions. Cryptography moves from the status of disabling some attacks, fixing the security shortcomings of current IT, to the status of **enabler**. Making possible new forms of distributed cooperative business and services. Cryptography is a disruptive technology.

1.1 Bitcoin and Key Management

In this paper we look at the question of key management in bitcoin and other crypto currencies. This question actually becomes important for interesting technical reasons.

1. First, bitcoin offers pseudonymity but poor anonymity, cf. [18, 13]. Using multiple keys helps to achieve better anonymity without really being a panacea.
2. In addition bitcoin uses a non-orthodox elliptic curve, while in most cases public keys are only revealed at the first spending transaction. Therefore it is recommended never to use the same key twice, in this way the attacker cannot know the public key in advance which makes key recovery attacks truly improbable; the attacker would need to compute an Elliptic Curve Discrete Log in matter of a few seconds cf. [10] in order to have a reasonable chance to double spend an existing quantity of bitcoins which was already spent.
3. The questions of key management are important in order to insure reliable backup of bitcoin wallets. If a wallet contains randomly generated keys, sooner or later it runs out of keys, and the backup does no longer allow to recover all the private keys and some bitcoins could be lost. The only solution known is to derive all keys in a deterministic way.
4. There is also a question of cold storage of bitcoins, for example in a safe. Using random keys leads to large volumes of keys which are very hard to manage in practice for humans. Companies which do cold storage at a large scale for numerous customers need again use deterministic key management solutions. It appears that billions of dollars are currently stored in such systems.

1.2 Holy Grail Bitcoin of Key Management: Audit Capability

In the historical process of development of deterministic key management solutions in bitcoin, we have seen another "cryptographer's dream" in operation. The idea that it could be possible to assure some level of transparency in bitcoin world without compromising security. The key question is the **audit capability**. For example in a financial institution or broker firm we would like to give some people, auditors, the capacity to see all the transactions without being able to spend the money. Similarly in a cold storage system we we would like to know how much money we have at any moment. this capability is also essential in e-commerce scenarios, where the server connected to the Internet which accepts payments must be able to generate public keys on the fly, cf. see Section 2. This audit capacity is actually the source of all the trouble and the source of most attacks on this paper. The bottom line is that this "audit-ability dream" is somewhat possible with current public key cryptography, and yet somewhat deficient, leading to numerous interesting attacks.

This paper is organized as follows. In Section 2 we study the question of deterministic private key management. In Section 3 we look at generic solutions which allow to derive private and public keys with the audit capability. In Sec. 3.1 we look at a specific solutions which are specified in BIP032 and in Sec. 3.3 we describe the basic privilege escalation attack. In Section 4 we discuss the traditional bad RNG attacks on ECDSA, show that such events have occurred in bitcoin, and introduce more sophisticated related random events. In Section 5 we describe new more general related random attacks.

In Section 6 we describe three advanced combination attacks. In Section 7 we describe three further attacks across two distinct domains which have no shared secrets. In Section 8 we show that these lead also to some 6 more analogous attacks in which instead of identical randoms, related randoms are generated by different users, and also describe a more general attack with an arbitrary number of domains. Finally in Section 9 we show another practical attack scenario which takes it even one stop further: that similar attacks also work in a variety of practical scenarios where less bad or related randoms occur, however some events additional in which the private keys overlap have also been detected.

2 Deterministic Private Key Management

In symmetric cryptography we use deterministic key derivation functions for a number of reasons.

1. They limit key exposure and they limit amount of data that the attacker can dispose of, for both traditional algorithmic and side-channel attacks.
2. It avoids expensive storage of too many keys, avoids large databases and the problem of updating them as new devices join the system.

Now for any system with public key cryptography, we can derive private keys in the same way, and then derive public keys from private keys on an individual basis, key per key. We call it Type 1 Wallets in bitcoin jargon.

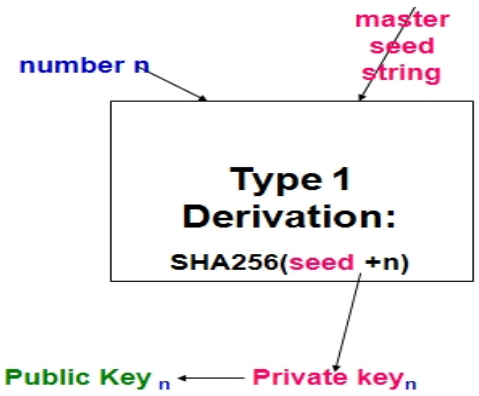

Fig. 1. The principle of Type 1 wallets.

Many bitcoin applications use Type 1 solutions and their widespread application is attributed to Greg Maxwell, Pieter Wuille and other bitcoin developers [23, 22, 16]. Type 1 solutions typically derive keys by applying SHA256 to a master seed with an added counter n.

Type 1 Wallets are secure (if SHA256 is secure) but not very practical. We would like to have audit capability: the capacity for auditors to see public keys but not private keys without again having all the problems implied by a large database of keys. This capacity is also needed in an e-commerce scenarios we would like a web site to be able to generate unique keys for receiving bitcoins for a large number of different transactions without being able to spend any of the bitcoin funds (web servers are a big target for hackers). It also increases anonymity of commercial transactions knowing that the bitcoin blockchain is public (web sites which accept credit card payments do not have this problem).

2.1 Type 2 Key Management Solutions

The crucial property sought for a wallet to be a Type 2 solution is as follows. We want to have 2 derivation functions such that the following diagram commutes. For example PK_3 should be the same whether it is computed by the Public derivation function or if it is computed from SK_3. In addition the public seed should NOT reveal the private seed or any of the private keys. All the 4 derivations on the diagram should be done with one-way functions.

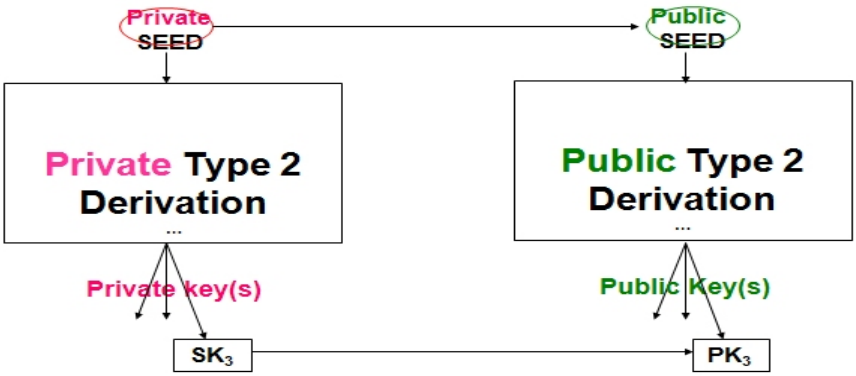

Fig. 2. The principle of Type 2 wallets: the two key derivation functions must be such that the diagram commutes and public keys PK_i obtained either way are the same.

So far it is possible to see that the goal is achievable. We can propose solutions which satisfy these requirements. The main idea [attributed to Greg Maxwell and Peter Wuille] is as follows, cf. [23, 22, 16]. Both derivation functions are essentially the same, however in the "Private Type 2 Derivation" function the private key is simply used to compute the corresponding public key. We will use two distinct algebraic properties of ECDSA keys in order to be able to obtain two such solutions for ECDSA keys specifically. In fact more advanced solutions exist which have also two additional properties:

1. They have additional long secrets beyond public/private keys. These additional secrets are called "chain" values, so that knowing all ECDSA keys still does not allow to derive further keys.
2. They work across several levels of further specializatiin/derivation (cf. Fig. 3, below), allowing very interesting and elegant key management solutions with multiple hierarchical security domains and sub-domains.
3. Similarly they allow to generate individual keys for individual transactions.

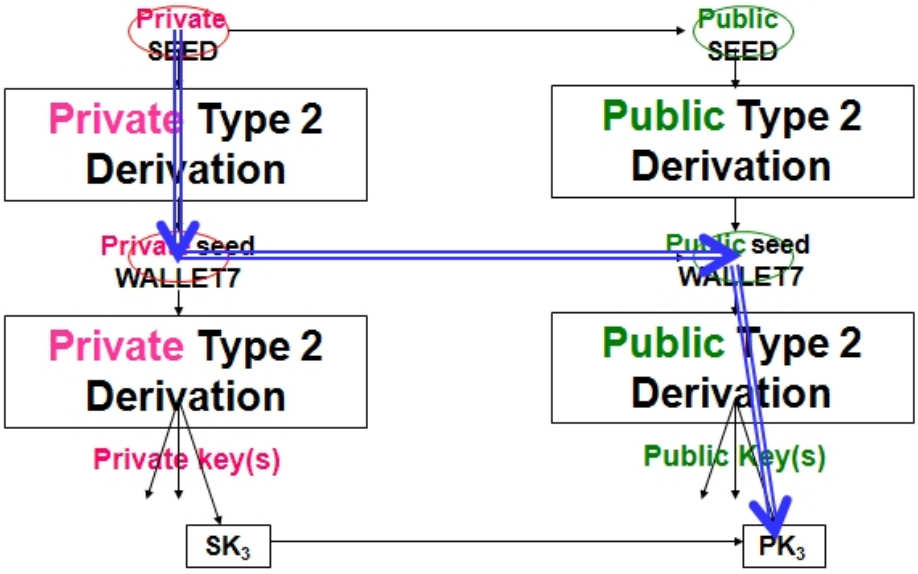

Fig. 3. The principle of hierarchical Type 2 wallets across several levels: the diagram should commute with all possible paths giving the same result.

Again it appears that the goal is achievable and we are going to describe some generic and one more specific solutions which seem to satisfy these requirements. We are going to need the following definitions:

Definition 21. We call a security sub-domain a set of all the nodes in the DIRECTED graph as represented in Fig. 3, with all the nodes and leaves BELOW one point/edge, in the sense of the arrows, which corresponds to what can be computed: lower level keys can be computed from higher level, and public seeds CAN be computed from private seeds.

Remark: The level of access on any user in an HD wallet system can be seen as a union of several security domains in the sense of Def. 21 above. However the possession of keys which give access to several domains can have unintended consequences, as a clever attacker will be sometimes able to gain access to additional security domains. See our *Privilege Escalation Attack* in Prop. 32 on page 11.

3 Generic Solutions

In this section we two generic solutions which works across an arbitrary number of levels. In this paper we only cover solutions with additional secrets (chain variables) so that seeds in Fig. 3 will now always be of specific form asymmetric key (left part) concatenated with a Chain variable C (right part). Both solutions are specific to ECDSA in the sense that a similar solution probably wouldn't exist if a different digital signature scheme with less special algebraic/homorphic properties was used in bitcoin. It is possible to see that both solutions could also work with traditional DSA-like signatures. In this paper we will use the additive notation (as in ECDSA) and ignore their DSA versions. All attacks in this paper are totally independent of the elliptic curve used.
We are going to need the following definition:

Definition 31. We assume a public key cryptosystem such that the private key can be derived in an efficient deterministic way from the public key (e.g. ECDSA). We call an *Extended Private Key* a concatenation of a private key and an additional secret x_C which will be called Chain variable.
We call an *Extended Public Key* a concatenation of a public key and an additional secret x_C which under right circumstances it will be the same value as above.
We call **Neutering** or $\mathcal{N}$ the operation which transforms an *Extended Private Key* into an *Extended Public Key* as follows: we derive the public key for the left part by EC base point multiplication, and we copy the Chain variable x_C, so we have

$$\mathcal{N}(k||x_C) = (k.G||x_C).$$

This operation $\mathcal{N}$ of **Neutering** is what is need to make diagrams such as in Fig. 3 commute. In all the solutions presented in this paper and on all pictures in this paper each of the horizontal arrows represents an application of $\mathcal{N}$ on extended keys with chain values.

3.1 Two Solutions

Interestingly, even though the problem is not completely trivial to solve, there two distinct major generic solutions to this problem known, both of which allow flexible audit capabilities. By flexible we mean that we can give auditors extended keys (cf. Def. 31) which allow audit of all operations in ANY security sub-domain as defined in Def. 21 above, and which works down across an arbitrary number of levels. We present both solutions in Fig. 4 and Fig. 5 below.

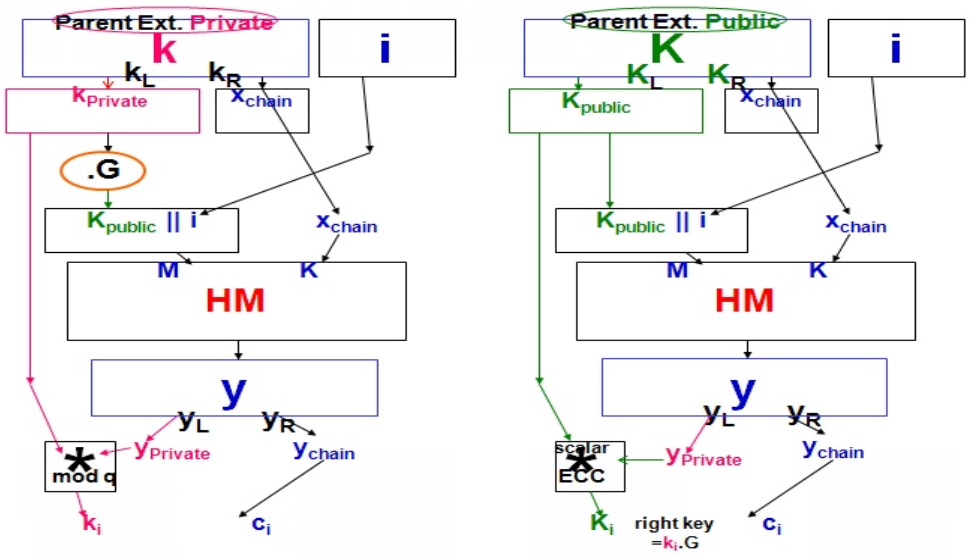

Fig. 4. A generic Solution1 using the EC scalar multiplication property, HM is an arbitrary OW function with 2 inputs called M and K. In practice HMAC can be used.

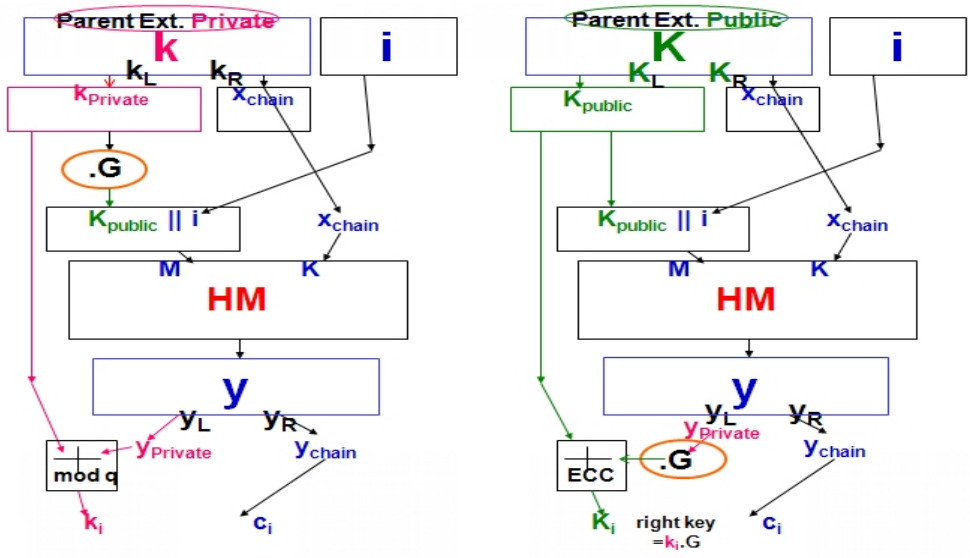

Fig. 5. A generic Solution2 using the EC homomorphic property, HM is an arbitrary OW function with 2 inputs called M and K. In practice HMAC can be used.

In all cases we denote by q be the order of the EC subgroup used. In bitcoin $q = 2^{256} - 2^{32} - 977$ we have. In both solutions if we assume that we apply $\mathcal{N}$ to the inputs on the left side and we will get the inputs

on the right side, the same relationship also holds for the two outputs, i.e. we can apply $\mathcal{N}$ to the outputs on the left side and we will get the same correct outputs on the right side. We refer to Proposition 31 and Corollary 31 below for a proof of correctness and applicability of these two solutions across an arbitrary number of levels as in Fig. 3. We have the following results:

Proposition 31. For both Solution1 and Solution2 with notations as in Fig. 4 and Fig. 5 if we have that

$$\mathcal{N}(k||x_C) = (K||x_C).$$

which means that $K = k.G$ and x_{chain} is the same on both sides, THEN then we obtain the right key with

$$\mathcal{N}(k_i, c_i) = (K_i, c_i).$$

or in other words we have $K_i = k_i.G$ and the same c_i is obtained on both sides.

Proof: The proof is obvious by following the identical values on the diagram in Fig. 4 and Fig. 5. In Solution1 in Fig. 4 the public key K_i on the right hand side is the correct key for k_i on the left because we have

$$k_i.G = (y_{Li} * k_L).G = y_{Li}.(k_L.G) = y_{Li}.K_L = K_i.$$

In Solution2 in Fig. 5 and in BIP032 in Fig. 6 the public key K_i on the right hand side is the correct key for k_i on the left because we have

$$k_i.G = (y_{Li} + k_L).G = y_{Li}.G + k_L.G = y_{Li}.G + K_L = K_i.$$

More generally for both solutions and by induction it is easy to see that:

Corollary 31. Both the solution of Fig. 4 and Fig. 5 can produce commuting diagrams across an arbitrary number of levels such as in Fig. 3 in which all directed paths produce exactly the same public keys.

Remark 1. Both solutions exploit two distinct properties on ECDSA keys. We have only two major solutions in Fig. 4/5 and it is NOT easy to produce similar solutions. It is not true that one could use an arbitrary group operation to combine y_L with k and produce other solutions[4].

Remark 2. Both solutions have been implemented and are very widely used in bitcoin community; in bitcoin exchanges, cold storage systems, wallet systems, electronic commerce solutions, trading solutions etc. The Solution1 is based on the initial proposal which has been used in various systems such as Electrum before it was changed on 30 April 2013 to implement Solution2 which is now standardized as part of BIP032 specification and more widely used. It is not clear that Solution2 was ever meant to be more secure than Solution1, however it was claimed to be faster and easier to implement [22]. In this paper was show that Solution2/BIP032 is NOT more secure than Solution1, and we show similar sort of attacks on both solutions.

[4] Quite possibly there would be NO solution at all if bitcoin has used a different sort of digital signature scheme (e.g. Lamport signatures [6]) which does not have special algebraic properties.

3.2 A Specific Solution: Details About BIP032

BIP032 is a specific application of Solution2 based on HMAC-SHA512.

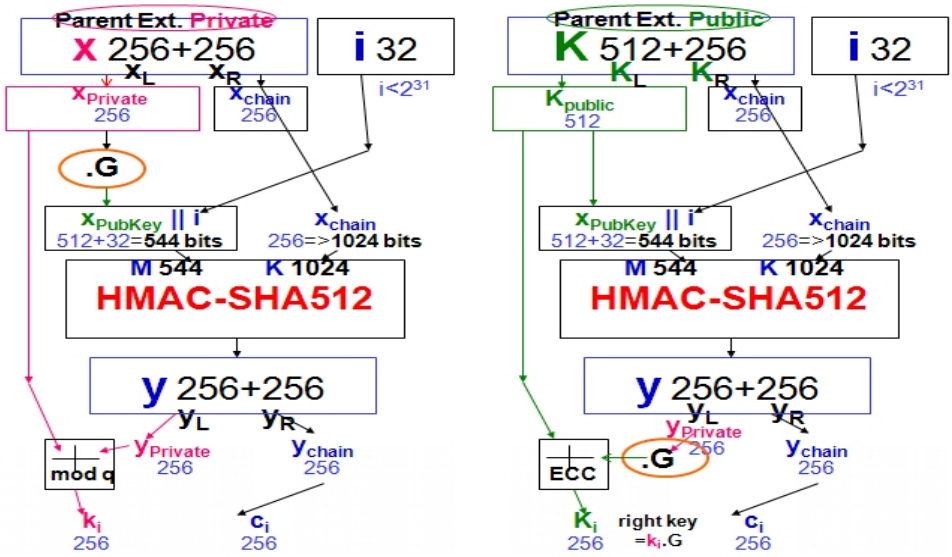

Fig. 6. Details of BIP032 public and private key derivation mechanisms known as CDK or Child Key Derivation functions, both work together.

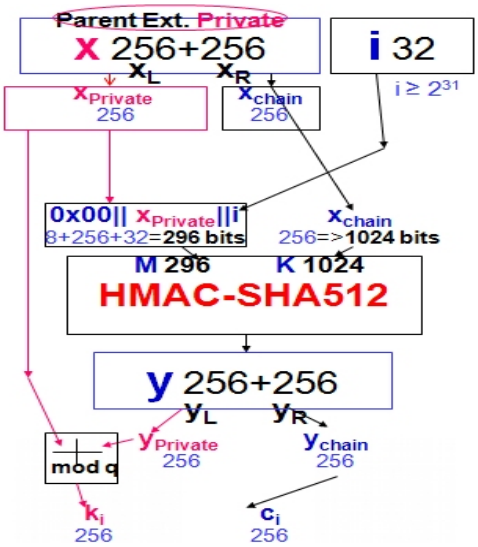

Fig. 7. The "Hardened" version of BIP032 private key derivation function when $i \geq 2^{31}$. This is version is back again a Type1 solution like on Fig. 1. Here the "compatible" public version does not exist and there is no way to derive public keys in bulk.

3.3 Basic Attacks Against The Two Solutions and BIP032

Both solutions admit a number of relatively simple attacks. These attacks can be considered as known and most have already been discussed in various bitcoin forums for at least one of the two versions, and are also mentioned by the author of BIP032 Pieter Wuille in both [22] and [23]. However there is no systematic study of these attacks and it was not clear if similar attacks always exist for both solutions and if they work across an arbitrary number of levels. All these attacks can be summarized in one single general privilege escalation attack. This is what we are to show now.

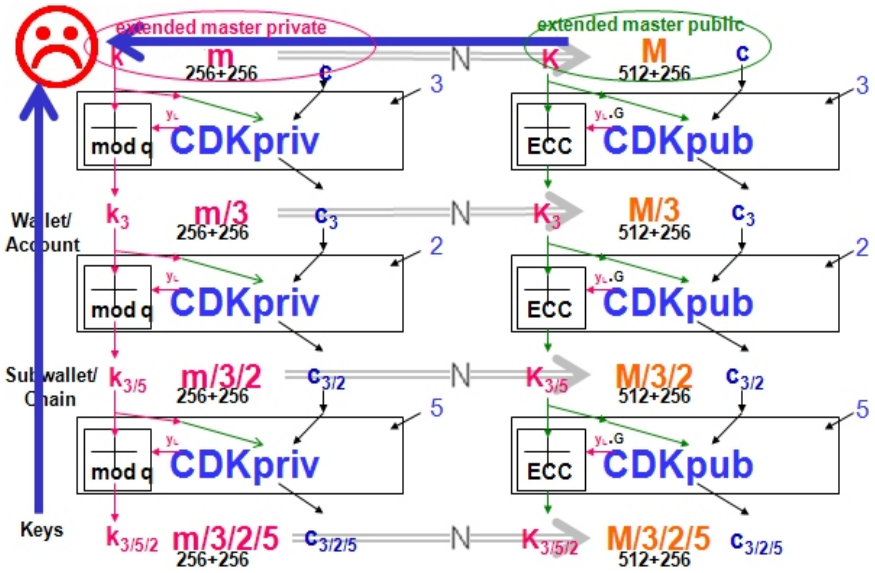

Fig. 8. One high-level extended public key and ANY private key at any level below combined allow to compromise the master private secret k and derive any other key.

Proposition 32 (Privilege Escalation Attack). For both solutions in Fig. 4/5 applied across an arbitrary number of levels we have: if the attacker disposes of the audit key K, x_c for one Security Domain as defined in Def. 21 AND if the attacker knows just one private key at any level one or several degrees below k, say he knows a private key k_I where I is a a set of 1 or more indices, THEN the attacker can compute the high level private key k at the level of the audit key known.
Therefore he can also compute any other private key and corresponding secret chain value k_J, c_J for any other arbitrary path J of length ≥ 1 below k and similarly also arbitrary public keys below K.

Proof: For simplicity we first assume there is only one layer of key derivation functions. We recall that q is the order of the EC subgroup used. With Solution1 one private key is of the form $y_1 * k \mod q$ and the other is $y_2 * k \mod q$. The auditor key K, x_c allows to compute both y_1, y_2 so he can compute k and $y_2 * k \mod q$.
With Solution2 or/and BIP032 one private key is of the form $y_1 + k \mod q$ and the other is $y_2 + k \mod q$. The auditor key K, x_c allows to compute both y_1, y_2 so he can compute k and $y_2 + k \mod q$. By induction we can go up an arbitrary number of levels as long as we have the audit extended key on these levels which allow us to compute all the y_i values needed.

Discussion. The privilege escalation attack of Prop. 32 above can be considered as a very serious attack, because a coalition of two people can compromise the whole system. However if all the digital signatures are done on the cold machines and in absence of an exploit or malware on the cold machine, the system remains secure. In this paper we describe many new attacks all of which operate under a weaker assumption and are such that the attacker does not need to run any code on the cold machine, he just needs to identify some bad randomness events.

4 Bad Randomness Attacks On Bitcoin

In this Section we summarize what is known about bad randomness attacks in bitcoin. In fact the full picture about these vulnerabilities have never been made public and it this section we disclose a number of facts and the relevant security and impact analysis which has not yet been published.

4.1 History Of Bad Randomness Attacks on Bitcoin

On 28 January 2013 Nils Schneider have reported that repeated randoms have occurred in the bitcoin data. He has found two bitcoin transactions which use the same random and published this fact on his blog [19]. The problematic transaction in question was from March 2012 and therefore the problem remained undetected for quite some time. Few days later Moore and Wustrow publish more detailed results [14]. They have examined some 11 million bitcoin transactions to date and found 126 repeated random values on 256 bits which appear in 316 different transactions. They found that most values on 256 bits were repeated only once. However some values were repeated a few times, and the value D47CE4C025C35EC440BC81D99834A624 875161A26BF56EF7 FDC0 F5D52F843AD1 which was already found by Schneider [19] was repeated more than 50 times In August 2013 we found on the Internet another file posted anonymously by a certain Greg, which contained 131 bad randoms. However already at this moment this file already was incomplete, we have found a few more such events at the time and we have reported some additional events inside the Code Breakers group at LinkedIn. At 30C3 conference in Germany on 28 Dec 2013 [5] Nadia Heninger have reported that they have identified a bitcoin user on the blockchain which has stolen some 59 BTC due to these bad randomness events which address is 1HKywxiL4JziqXrzLKhmB6a74ma6kxbSDj, see also [21]. To this day this money remains on the blockchain and has not been spent.

4.2 A Recent Outbreak of Bad Random Events

Most of the events form 2013 have happened in May 2013, cf. Fig. 9, and then have almost entirely disappeared, presumably because the Anroid applications affected by the problem have been reportedly updated (including our own) and applications have transferred bitcoins automatically to new accounts with fresh bitcoin addresses.
A recent massive outbreak of bad random events have occurred in May 2014, cf. [21] and Fig. 9 below.
Here is an example of a very recent bad random from 28 September 2014: 0f25a7cc9e76ef38c0feadcfa5550c173d845ce36e16bde09829a3af57097240.
This random r appears 8 times in block 322925 and has been used with different private keys so that it is not obvious if bitcoins can be stolen, see Section 4.3 below. However if any of the HD Wallet solutions have been used to generate at least two of these keys, for sure bitcoins can be stolen, and not only from these 8 accounts, but from potentially a larger domain with a lot more accounts, cf. our later Proposition 63.

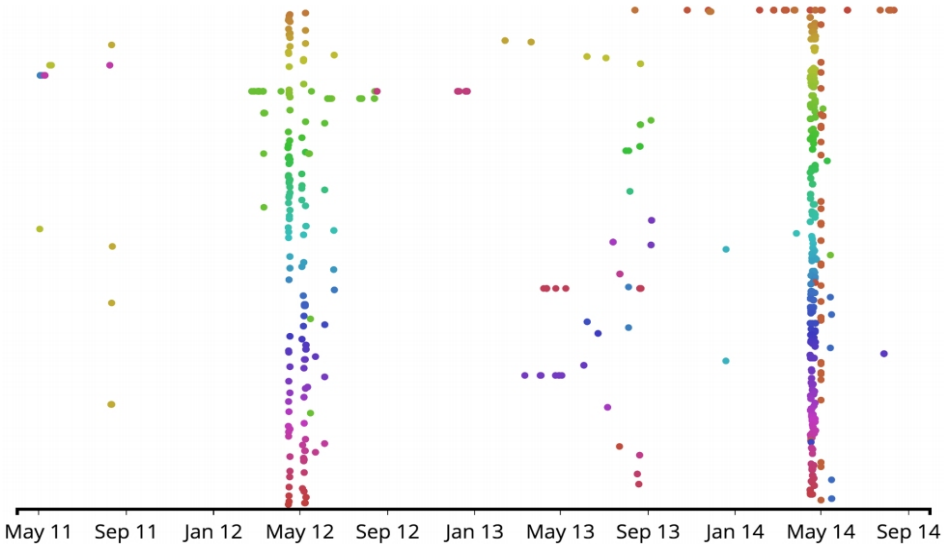

Fig. 9. Bad randomness events in bitcoin blockchain, the sending address is represented on the vertical axis. Short horizontal lines suggest that some vulnerable applications have used the same address many times in a row, vertical lines suggest that other applications have used many randomly generated addresses.

4.3 Basic Attacks with Bad Randomness and ECDSA

In order to explain better the events, we need to explain what are the exact implications of using the same random twice in a DSA/ECDSA-like signature. Bad randomness in various industrial systems is a widely known problem, see [7, 12, 11, 19, 21]. In this paper we need first to recall basic attacks more or less in the line of specific incidents already studied in previous works [1, 11, 19, 21] and later we will propose new more advanced attacks. We recall that:

Definition 41 (ECDSA Signature).
If the integer $0 < k < q - 1$ is the ECDSA private key, and q is the order of the base point G, an ECDSA signature is computed as follows. We pick a random non-zero number $a \mod q$ and the signature of m is the pair r, s with

$$r = (a.G)_x; \quad s = (H(m) + kr)/a \mod q$$

It is easy to see that we have the following two results:

Proposition 41 (Random Recovery in ECDSA). If the attacker knows the original random a, one message and one valid signature m, r, s, he can compute the private key k with the following formula:

$$k = (sa - H(m))/r \mod q.$$

Proposition 42 (Key recovery From Random). From one valid signature m, r, s and the private key k one can later recover the original random a as follows:

$$a = (H(m) + kr)/s \mod q.$$

Therefore:

Proposition 43 (Attack on ECDSA with Bad RNG). If the same random a is used in two different ECDSA signatures m_1, r, s_1, and m_2, r, s_2, each of the two signers can compute the private key of the other signer.

Proof: We recover a due to Prop. 41 and then we apply Prop. 41.

Remark 1. It is possible to see that for the public (people other than the 2 signers) and from the public data m_1, r, s_1, and m_2, r, s_2 alone, it is **NOT** possible to compromise the private keys of the two users
Remark 2. This changes if we know the source code of the RNG and have a chance to reproduce a. Then we can recover both private keys (cf. Prop 42).
Remark 3. In bitcoin network the compromise of one private key allows to steal all the money in one specific account. They can also be stolen at any moment in the future, i.e. moneys which were not there earlier. Money can be stolen even the RNG has been fixed and produces good random numbers later on, however some senders still use the old public bitcoin address. This is very hard to avoid: potentially anyone can send money to an older address.

4.4 Special Cases

There is an interesting special case.

Proposition 44 (Bad RNG ECDSA Same User Attack). If two digital signatures are produced with the same private key AND the same random, then given m_1, r, s_1, and m_2, r, s_2, **anyone** can compute the private key of that person at any moment.

Proof: We recall how it is done (cf. also [19]). We have

$$(s_1 a - H(m_1)) = ak = (s_2 a - H(m_2)) \mod q$$

This gives $a = (H(m_1) - H(m_2))/(s_1 - s_2) \mod q$ and we can recover the private key k using Prop. 42.
There are also cases in which the keys acan be recovered by anyone, basically if such events happen many times. For example we have:

Proposition 45 (Attack with Two Bad Random Events). If two identical randoms a and b are used just once by two different users (each user needs to use both a and b, then anyone can recover the private keys for both users.

Proof: This is a special case of our later Prop. 71.

4.5 Our Bitcoin Network Scan - Classification of Events

In our scan there are three basic sorts of events:

1. The same random was used by the same person with the same private key in the same transaction. In this case following Prop. 44 above **anyone** can compute the private key of that person at any moment and steal his coins (also in the future). This seems unlikely but has already happened in the original case from March 2012 cf. [19] and is also the case for the majority of 148 events.
2. The same random is used by two different people. In this case following Prop. 43 above each signer can compute the private key of the other signer and steal all his coins, but there is no immediate[5] threat of other people stealing bitcoins.
3. There is a mix of 1. and 2. The same random has been used at least 3 times once it was used twice by the same person, and sometimes it was also used by other people. Then Prop. 44 above the random a can be recovered and then following Prop. 42 the private keys for the other users can also be compromised.
4. The same random was used by the same person with different private keys, which is the same as case 1. above however that person already knows all the keys and there is no harm.

The crucial remark is that in case 2. attackers who are not any of the two people can do nothing. In this paper we present a number of combination attacks where this will nevertheless work and lead to a compromise of a whole large security domain (all private keys will be recovered).

4.6 Which Applications Are Vulnerable

Some analysis of which applications are vulnerable can be found at the end of [21]. The analysis is very disturbing: it appears that the bitcoin core client has NOT yet finalized the process of several security upgrades which propose a solution to this problem (deterministic random generation, cf. [17, 21]) event though a first solution was already submitted in January 2013. It appears also that Blockchain.info wallet is also unsafe as of today.

[5] This will change if this sort of thing happens more than once, see for example Prop. 45 above and also point 3. and also cf. advanced combined attacks in Section 6 and 7 below.

5 Advanced Attacks With Related Randoms

In the well known research on bad RNG in public key cryptography and TLS connections [11] there are two sorts of events:

1. Events when two primes generated by two different users are identical. Thousands of such cases have been identified in [11] and we have identified a few hundreds more such events in our own scan.
2. Events where two primes are NOT identical but there are shifted by a number of bytes.

To the best of our knowledge such events 2. were NOT yet identified in the public literature in TLS connections. We expect that a very large number of such events have occurred in the real life. This is based on our own experimentations done at UCL: recovering some 8000 primes which have been obtained by our students in the same way as in [11]. We have then discovered that 10 of these primes had at least 200 bits in common with another prime in the set with various shifts, cf. [1].
This is of course just a tip of the iceberg: there are many other primes in this dataset which share long strings of bits with other primes, and we have NOT yet been able to discover them because they belong to primes which have not yet been factored with the method of [11]. Some related research can be found in [4].

5.1 Related Randoms In Bitcoin

What happens if two related randoms are generated by two different bitcoin clients? Again depending on how big is the overlap this is going to allow each of these people to steal other people's bitcoins. Let us assume that two random a and b are such that b starts at a position 8 bits later with the same RNG setup. again this sort of event has happened in the real life. Then both randoms a and b can be recovered as follows:

1. We have:
$$b + c \cdot 2^{256-8} = 256 \cdot a + d$$
Where c and d are very small 8-bit integers with can be guessed
2. Therefore if r_1, s_1 is a digital signature obtained with a, and r_2, s_2 is the one obtained by another user with b, we have on the elliptic curve:
$$r_2 + c \cdot 2^{256-8}.G = 256 \cdot r_1 + c.G$$
3. This equation allows to find the right pair c, d in time of 2^{16} EC operations.

Again as with Prop. 43, in principle this does not YET allow anyone to steal bitcoins of these two users. For now only these two users can steal some bitcoins. This will however change if HD Wallets with any Solution1 or 2 are also used at the same time, see Section 8 and 9 below.

Remark. In a future release of this paper we are going to reveal a much more efficient and more practical method for finding such relations.

6 Advanced Combination Attacks

This is the main part of this paper. It contains several new attacks never published before and never claimed before. The new attacks are **combination attacks** which combine distinct vulnerabilities we has studied so far (cf. Section 3.3 and Section 4). They allow to recover private keys which none of the two vulnerabilities alone would allow to hope to recover. In fact typically it can lead to a total compromise of a given system (all private key revealed), or to a partial compromise (a whole sub-domain). We describe several such attacks which work across an arbitrary number of levels of key derivation and in a variety of practical scenarios. The first one is going to be pretty trivial.

Proposition 61 (Combined Attack Bad RNG+Solution1/2). If the attacker knows the *Extended Public Key* $K||x_C$ for a whole system or for any security sub-domain as defined in Def. 21 AND if two identical randoms were used by a single user anywhere in the system, then he can recover ALL private keys in this sub-domain as well as the master private key for the same whole sub-domain.

Proof: We combine Prop. 44 page 15 and Prop. 32 page 11.
This was easy, however we cannot hope that such events will happen. What if the same random is used by two different users? This is more probable. Here we will have a more serious attack which will really show how fragile HD Wallet solutions are.

Proposition 62 (Combined Attack On Solution1). If the attacker knows the *Extended Public Key* $K||x_C$ for a whole system or for any security sub-domain as defined in Def. 21 AND if two identical randoms are used just once by any pair of users in this domain, and if the relative paths I_1 and I_2 of these users in the domain are known[6], then he can recover ALL private keys in the domain as well as the master private key for the same whole sub-domain (or for the whole system).

Proof: We proceed as follows. From $K||x_C$ and I_1 the attacker can compute all the multipliers y_L which are applied to the private key along the path in one or more steps such as in Fig. 4. Let y_1 be the product of these multipliers. Similarly let y_2 be the same quantity for the second path I_2. Then the two private keys are ky_1 and ky_2 with unknown k. We have:

$$\begin{aligned}
as_1 &= (H(m_1) + rky_1) \mod q \\
as_2 &= (H(m_2) + rky_2) \mod q \\
(H(m_1) + rky_1)s_2 &= (H(m_2) + rky_2)s_1 \mod q \\
k(ry_1s_2 - ry_2s_1) &= H(m_2)s_1 - H(m_1)s_2 \mod q \\
k &= \frac{H(m_2)s_1 - H(m_1)s_2}{ry_1s_2 - ry_2s_1} \mod q
\end{aligned}$$

[6] As in Section 3.3 I can be any sequence of 1 or more indices, for example 3/5, from the unknown *Extended Private Key* $k||x_C$

The last equation allows to recover the top-level private key k for this domain. This completes the proof.

We have a similar attack for Solution2 of Fig. 5 BIP032 in particular.

Proposition 63 (Combined Attack On Solution2/BIP032). If the attacker knows the *Extended Public Key* $K||x_C$ for a whole system or for any security sub-domain as defined in Def. 21 AND if two identical randoms are used just once by any pair of users in this domain, and if the relative paths I_1 and I_2 of these users in the domain are known then he can recover ALL private keys in the domain as well as the master private key for the same whole sub-domain (or for the whole system).

Proof: Again From $K||x_C$ and I_1 the attacker can compute all the differences $y_L \mod q$ which are applied to the private keys along the path as in Fig. 5. Let y_1 be the sum of these differences $\mod q$ and let y_2 be the same sum quantity for the second path I_2. Then the two private keys are $k + y_1$ and $k + y_2$ with unknown k. We have:

$$\begin{aligned} as_1 &= (H(m_1) + rk + ry_1) \mod q \\ as_2 &= (H(m_2) + rk + ry_2) \mod q \\ (H(m_1) + rk + ry_1)s_2 &= (H(m_2) + rk + ry_2)s_1 \mod q \\ kr(s_2 - s_1) &= H(m_2)s_1 - H(m_1)s_2 - r(y_1s_2 - y_2s_1) \mod q \\ k &= \frac{H(m_2)s_1 - H(m_1)s_2 - r(y_1s_2 - y_2s_1)}{r(s_2 - s_1)} \mod q \end{aligned}$$

Again the last equation allows to recover the top-level private key k.

7 Advanced Combination Attacks Across Domains

In this section we study even more advanced attacks. In these attacks we will attack two totally independent HD wallet key management solutions. For example that there are two bitcoin exchanges with cold storage. If by unhappy accident a certain random is used twice in one single domain, we already know that this domain is compromised. Now what happens if the same random is used twice but in two different domains. In principle there is no hope for any attack. Now let us assume that it happens twice with two different unknown randoms a and b. Then there is an attack! First we look at Solution1. We have

Proposition 71 (Combined Attack Solution1 Two Domains). If the attacker knows the audit keys from two different domains with Solution1 AND if two identical randoms a and b are used just once by one user in each domain (four users total) and if we know relative paths of all users, then the attacker can recover ALL private keys and both master private keys in both domains.

Proof: Here the four private keys are ky_1, $k'y_2$, ky_3, $k'y_4$, with unknown k and unknown k' but the four values y_i can be computed from the audit keys for known paths. We have:

$$
\begin{aligned}
as_1 &= (H(m_1) + r_a k y_1) \mod q \\
as_2 &= (H(m_2) + r_a k' y_2) \mod q \\
bs_3 &= (H(m_3) + r_b k y_3) \mod q \\
bs_4 &= (H(m_4) + r_b k' y_4) \mod q \\
ky_1 s_2 - k' y_2 s_1 &= \frac{H(m_2)s_1 - H(m_1)s_2}{r_a} \mod q \\
ky_3 s_4 - k' y_4 s_3 &= \frac{H(m_4)s_3 - H(m_3)s_4}{r_b} \mod q
\end{aligned}
$$

This gives us 2 linear equations with 2 unknowns k and k' which allows to compute BOTH master private keys(!). All the other values are known.

A similar attack holds for Solution2/BIP032 but the computations are different.

Proposition 72 (Combined Attack Solution2/BIP032 Two Domains). If the attacker knows the audit keys from two different domains with Solution2 AND if two identical randoms a and b are used just once by one user in each domain (four users total) and if we know relative paths of all users, then the attacker can recover ALL private keys and both master private keys in both domains.

Proof: Here the four private keys are $k + y_1$, $k' + y_2$, $k + y_3$, $k' + y_4$, with unknown k and unknown k' but the four values y_i can be computed from the audit keys for known paths. We have:

$$
\begin{aligned}
as_1 &= (H(m_1) + r_a k + r_a y_1) \mod q \\
as_2 &= (H(m_2) + r_a k' + r_a y_2) \mod q \\
bs_3 &= (H(m_3) + r_b k + r_b y_3) \mod q \\
bs_4 &= (H(m_4) + r_b k' + r_b y_4) \mod q \\
ks_2 - k' s_1 &= \frac{H(m_2)s_1 - H(m_1)s_2}{r_a} - y_1 s_2 + y_2 s_1 \mod q \\
ks_4 - k' s_3 &= \frac{H(m_4)s_3 - H(m_3)s_4}{r_b} - y_3 s_4 + y_4 s_3 \mod q
\end{aligned}
$$

This also gives us 2 linear equations with 2 unknowns k and k' which allows to compute BOTH master private keys(!).

7.1 Heterogenous Domains

Furthermore:

Proposition 73 (Attack Across Domains Using Different Key Management Solutions). A similar attack exists if one bitcoin key management system uses Solution1 and another uses Solution2 or BIP032.

Proof: Here is how all the private keys can be recovered in this case:

$$\begin{aligned} as_1 &= (H(m_1) + r_a k y_1) \mod q \\ as_2 &= (H(m_2) + r_a k' + r_a y_2) \mod q \\ bs_3 &= (H(m_3) + r_b k y_3) \mod q \\ bs_4 &= (H(m_4) + r_b k' + r_b y_4) \mod q \\ k y_1 s_2 - k' s_1 &= \frac{H(m_2)s_1 - H(m_1)s_2}{r_a} + y_2 s_1 \mod q \\ k y_3 s_4 - k' s_3 &= \frac{H(m_4)s_3 - H(m_3)s_4}{r_b} + y_4 s_3 \mod q \end{aligned}$$

Again both master private keys can be determined from the last 2 equations which will compromise all the keys in both domains.

7.2 Discussion, Hardened Option In BIP032

It is important to recall that in all the cross-domain attacks in this Section 6 the different domains do NOT initially share any keys, yet at the end ALL keys in ALL the domains are recovered.
Interestingly the BIP032 key management standard has an interesting option of "hardened" key derivation function which is applied if any index $i > 2^{31}$. If we apply these key derivation functions we obtain distinct domains with independent extended public keys, a near-equivalent of truly independent[7] domains. then we just treat these as distinct domains and all the three attacks in this Section 6 still do apply. Therefore it is possible to claim that the hardened option in BIP032 is **NOT yet an ideal solution** against the attacks described in this paper. Moreover the hardened option options makes key management against more difficult and makes it necessary for auditors to hold many more keys.

7.3 Future Research

Many more attacks such as described in this paper exist. It is possible to see that if we have an arbitrary number of bitcoin exchanges or cold storage solutions each of which uses an independent master seed, if we dispose of all the audit keys (i.e. *Extended Public Keys* for all the exchanges, and if repeated randoms can be obtained in operation for some pairs of exchanges, and if some sub-branches of the BIP032 system are hardened, after a small number of events of this type the master keys of **ALL** domains can be computed and all bitcoins in ALL the exchanges can be stolen.

[7] Their private keys are still related BUT attacks such as *Privilege Escalation Attack* of Prop. 32 are no longer possible.

8 Generalized Combined Attacks with Related Randoms

There are many further improvement attacks which will NOT by immediately visible on the blockchain. Basically it is possible to see that all the 3 previous attacks in Section 6 and the 3 further attacks in Section 7 can also be generalized to work without a single repeated random.
Instead bad randoms need to be shifted by a small shift, and such pairs of related bad randoms can be recovered in an offline stage as described in Section 5.1 from the public keys. Then after the offline stage, we obtain explicit linear equations with known coefficients c and d such as in 5.1 on the private keys. With these linear relations we can apply all the 5 attacks form Section 6 and their generalizations to more distinct security domains, and each time we will need less linear equations such as produced in the last two formulas in Section 7.1, because we already have additional linear equations. As soon as the rank of our linear system become full, all the private keys for ALL security domains involved can be recovered.
Attacks with related random events are particularly scary. With a larger shift between two randoms a and b which are not visible to the attack, it becomes more probable that they actually occur in the real life and yet increasingly harder to detect. We cannot claim that we have found all such events. It will take month, maybe years an potentially much longer to detect such security incidents in the public blockchain. Therefor these attacks are particularly scary. It is possible to see that no cold storage system using BIP032 and a randomly generator can claim that they are secure. There is no guarantee that all their bitcoin are NOT stolen today or tomorrow using one of our attacks. However it is possible to see that if they apply a deterministic method such as RFC6979, cf. [17, 21], none of the advanced combination attacks in this paper will work (in contrast the simpler coalition attack of Prop. 32 will still work).

9 Additional Attacks: Mix of Related Randoms and Related Keys

The attacks of Section 8 can be taken one stop further under different assumptions. Similar attacks also work in a variety of practical scenarios where less bad randoms which relate randoms occur. Then we can also generate additional linear equations with coefficients which may have very small entropy on the private keys, and such relations can be efficiently discovered in an offline stage in the same way as in Section 5.1. Again as above, this leads to even more linear relations on the private keys which can be recovered in an earlier stage of the attack.
We can mix the equations obtained from identical randoms, those which come from related randoms, those from related keys, and also simply those which come from identical keys. All these sorts of events are plausible and have happened hundreds of times in real life situations, some as recently as September 2014, see Section 4. They have a cumulative

effect. Again we can apply all the 5 attacks form Section 6 and their generalizations to more distinct security domains and steal all bitcoin for ALL security domains involved as soon there was a sufficient number of bad related random events of these 4 distinct sorts.

10 Additional Scenarios with an Insufficient Number of Problematic Events

Another practical scenario is that both randoms and keys have been generated using the same RNG with a shift. Then again it is possible to see that all the same attacks in Section 6, Section 7 and their generalizations to more distinct domains can be executed.
Finally, it is interesting to understand what happens WHEN in a general scenario with say 5 security domains and 4 bad random events identified by the attacker in the blockchain, after long and painful brute force exploration of plausible relations based on some heuristics, for example which users are more likely to use the same software setup and use the same randomness, the attack does NOT yet have enough linear equations to recover all the private keys. Then our attack still allows at least some people in the system, who do have additional keys, like ordinary users[8] who will be able to steal other people's bitcoins not only in the system in which they belong, but also from other distinct security domains.

[8] The may have just one private key for one transaction, or may have recorded the seed used in their RNG in just one spending transaction, which is sufficient to recover a private key, cf. Prop. 42.

11 Conclusion

In this paper we have looked at the question of security of current key management solutions standardized in bitcoin BIP032 and their earlier variants. It appears that billions of dollars are stored in such systems worldwide. In such systems private keys are stored on "cold" machines which are never connected to the Internet. In practice Type2 HD Wallet solutions (both current Solution2/BIP032 and another earlier Solution1) are maybe the only way known to build cold storage, exchange/trading, electronic commerce and other business systems which are "manageable" in practice. They avoid the explosion in the number of keys and managing updates if new users/keys are added to the system.
Unhappily such as systems are quite centralized and super fragile. They have high-level audit keys which are typically stored and highly available on hot machines or are available for certain people: auditors, servers which deal with accepting payments e.g. in electronic commerce, etc. These audit capabilities create a lot mathematical dependencies between various keys. Now if only there are some more such dependencies due to other reasons such as bad randomness or other incidents at operation in some remote corner of the system, everything collapses very badly. Our attacks allow to recover private keys which none of the earlier attacks in isolation could hope to recover. In this paper we show that both older Solution1 and newer Solution2/BIP032 are vulnerable to advanced combination attacks. Classical bad random attacks typically concern only few users which are very lucky and can steal other people's bitcoins. In our new attacks all bitcoins in cold storage over whole large security domains can be stolen, without executing any exploit or any malicious code on the cold machine, Our results generalize to an arbitrary number of distinct security domains. In certain circumstances all bitcoins in cold storage and at **each** domain can be compromised. Or certain users in one domain can help to steal keys in another domain, cf. Section 10.
Attacks with related randoms are particularly scary, cf. Section 8. Such attacks have not yet been studied in bitcoin literature. They are not discovered easily, it is more probable that such events occur in the real life than simpler events we already discovered, and yet they are much LESS likely to be detected in time, before it is too late.
The impact of this paper on current bitcoin storage systems is hard to evaluate. On the one hand side we can always hope that there wasn't enough bad random events inside the domains using BIP032 to be compromised. On the other hand unless they apply a deterministic method such as RFC6979 from [17, 21], which is a definite solution for most attacks, no existing bitcoin cold storage system using BIP032 should feel too confident to be secure against the attacks described in this paper.

Open Problems. An interesting question is whether there exist a key management scheme for ECDSA with flexible audit capabilities such that the attacks such as described in this paper would not work. If it exists, the current bitcoin key management standard BIP032 should be replaced by a new standard.

References

1. E. Alashwali, *Cryptographic Vulnerabilities in Real-Life Web Servers*, Master thesis, M.Sc. in Information Security, University College London, 31 August 2012.
2. Adam Back: *Hashcash - A Denial of Service Counter-Measure*, `http://www.hashcash.org/papers/hashcash.pdf`, August 2002.
3. Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun: *Bitter to Better : How to Make Bitcoin a Better Currency*, In Financial Cryptography and Data Security, FC'12, Springer, 2012.
4. Daniel J. Bernstein, Yun-An Chang, Chen-Mou Cheng, Li-Ping Chou, Nadia Heninger, Tanja Lange, Nicko van Someren: *Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild,* In Asiacrypt 2013, pp. 341-360. Slides: http://www.hyperelliptic.org/tanja/vortraege/20131030.pdf
5. Daniel Bernstein, Nadia Heninger, Tanja Lange: *2013 - The Year In Crypto,* Presentation at 30th Chaos Communication Congress [30c3], Congress Centrum Hamburg, Germany, 28 December 2013 `http://www.internet-history.info/media-library/mediaitem/1829-the-year-in-crypto-30c3.html`
6. Vitalik Buterin: *Bitcoin Is Not Quantum-Safe, And How We Can Fix It When Needed,* 30 July 2013, `http://bitcoinmagazine.com/6021/bitcoin-is-not-quantum-safe-and-how-we-can-fix/`.
7. Nicolas Courtois, Daniel Hulme, Kumail Hussain, Jerzy Gawinecki, Marek Grajek: *On Bad Randomness and Cloning of Contactless Payment and Building Smart Cards*, In IWCC 2013, International Workshop on Cyber Crime, San Francisco, USA, 24 May 2013. Slides: `http://www.nicolascourtois.com/papers/buildtransc_rng_sf_22052013.pdf`
8. Nicolas T. Courtois, Pinar Emirdag and Daniel A. Nagy: *Could Bitcoin Transactions Be 100x Faster?* in proceedings of SECRYPT 2014, 28-30 August 2014, Vienna, Austria.
   May 2014,
9. Wei Dai: *B-Money Proposal,* 1998, `http://www.weidai.com/bmoney.txt`
10. Christian Decker, Roger Wattenhofer: *Information propagation in the bitcoin network,* 13-th IEEE Conf. on Peer-to-Peer Computing, 2013.
11. N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, *Minding Your Ps and Q's: Detection of Widespread Weak Keys in Network Devices*, 2012, `https://factorable.net`. Accessed: 23-Jul-2012.
12. A. Lenstra, J. Hughes, M. Augier, J. Bos, T. Kleinjung, and C. Wachter: *Ron was wrong, Whit is right*, 2012. At `http://eprint.iacr.org/2012/064.pdf`.
13. Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage: *A Fistful of Bitcoins: Characterizing Payments Among Men with No Names*, In IMC 2013.
14. Jonathan Moore, Eric Wustrow: *Bitcoins and entropy*, `https://plus.google.com/u/0/106313804833283549032/posts/X1TvcxNhMWz`

15. Satoshi Nakamoto: *Bitcoin: A Peer-to-Peer Electronic Cash System,* At `http://bitcoin.org/bitcoin.pdf`
16. Satoshi Nakamoto *et al.*: *Bitcoin QT,* the original and the most prominent bitcoin software distribution which implements a full peer-to-peer network node. Originally developed by Satoshi Nakamoto, core developers are Satoshi Nakamoto, Gavin Andresen, Pieter Wuille, Nils Schneider, Jeff Garzik, Wladimir J. van der Laan and Gregory Maxwell. Available at `http://bitcoin.org/en/download` with source code at `https://github.com/bitcoin/bitcoin`.
17. Thomas Pornin, *Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA),* IETF, August 2013, `http://tools.ietf.org/html/rfc6979`
18. Dorit Ron and Adi Shamir: *Quantitative Analysis of the Full Bitcoin Transaction Graph,* In Financial Cryptography and Data Security 2013, preprint available at `http://eprint.iacr.org/2012/584`.
19. Nils Schneider: *Recovering Bitcoin private keys using weak signatures from the blockchain,* Blog entry, 28 January 2013, `http://www.nilsschneider.net/2013/01/28/recovering-bitcoin-private-keys.html`.
20. Technical specification of the bitcoin protocol, `https://en.bitcoin.it/wiki/Protocol_specification`
21. Filippo Valsorda: *Exploiting ECDSA Failures in the Bitcoin Blockchain,* slides presented at Hack in the Box Conference 2014, HitbSecConf2014, Kuala Lumpur, Malaysia, on 15 October 2014, at `http://conference.hitb.org/hitbsecconf2014kul/materials/D1T1-FilippoValsorda-ExploitingECDSAFailuresintheBitcoinBlockchain.pdf`
22. Pieter Wuille: the official specification of BIP032, 15 Jan 2014, `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki`
23. Pieter Wuille: presentation abut BIP032 at Bitcoin 2013 Conference San Jose, CA, May 2013, `https://www.youtube.com/watch?v=WcnMjkc31Fs`