WHEN THE ”CRYPTO” IN CRYPTOCURRENCIES BREAKS: BITCOIN SECURITY UNDER BROKEN PRIMITIVES

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

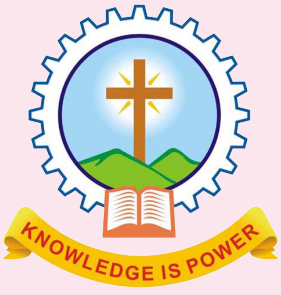
COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

SHILPA SREEDHAR K



Department of Computer Science & Engineering
Mar Athanasius College Of Engineering Kothamangalam

WHEN THE ”CRYPTO” IN CRYPTOCURRENCIES BREAKS: BITCOIN SECURITY UNDER BROKEN PRIMITIVES

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

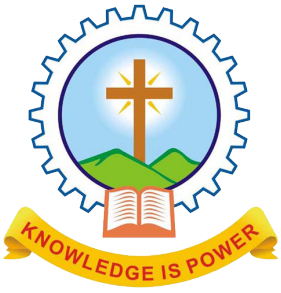
COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

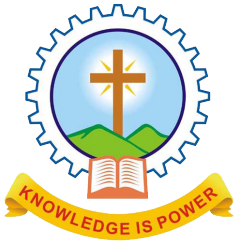
Submitted By

SHILPA SREEDHAR K



Department of Computer Science & Engineering
Mar Athanasius College Of Engineering Kothamangalam

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
KOTHAMANGALAM



CERTIFICATE

This is to certify that the report entitled **When the "Crypto" in Cryptocurrencies Breaks : Bitcoin Security Under Broken Primitives** *submitted by* **Ms. SHILPA SREEDHAR K**, *Reg.No.* **MAC15CS054** *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer science and Engineering from APJ Abdul Kalam Technological University for December 2018 is a bonafide record of the seminar carried out by him under our supervision and guidance.*

.....	.....	.....
Prof. Joby George	**Prof. Neethu Subash**	**Dr. Surekha Mariam Varghese**
Faculty Guide	***Faculty Guide***	***Head of the Department***

Date: Dept. Seal

ACKNOWLEDGEMENT

First and foremost, I sincerely thank the God Almighty for his grace for the successful and timely completion of the seminar.

I express my sincere gratitude and thanks to Dr. Solly George, Principal and Dr. Surekha Mariam Varghese, Head Of the Department for providing the necessary facilities and their encouragement and support.

I owe special thanks to the staff-in-charge Prof. Joby George , Prof. Neethu Subash and Prof. Joby Anu Mathew for their corrections, suggestions and sincere efforts to co-ordinate the seminar under a tight schedule.

I express my sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar.

Finally, I would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to me by my dear friends during the preparation of the seminar and also during the presentation without whose support this work would have been all the more difficult to accomplish.

ABSTRACT

Digital currencies such as Bitcoin rely on cryptographic primitives to operate. However, past experience shows that cryptographic primitives do not last forever. Increased computational power and advanced cryptanalysis cause primitives to break and motivate the development of new ones. It is therefore crucial for maintaining trust in a crypto currency to anticipate such breakage. Here the first systematic analysis of the effect of broken primitives on Bitcoin is explained. The analysis shows the ways in which Bitcoins core cryptographic building blocks can break and the subsequent effect on the main Bitcoin security guarantees. The analysis reveals a wide range of possible effects depending on the primitive and type of breakage. The effect ranges from minor privacy violations to a complete breakdown of the currency. The results lead to several suggestions for the Bitcoin migration plans and insights for other cryptocurrencies in case of broken or weakened cryptographic primitives and relate Bitcoins security model to that of other currencies.

Contents

List of Figures

List of Abbreviations

RIPEMD	RACE Integrity Primitives Evaluation Message Digest
SHA256	Secure Hash Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
POW	Proof of Work
P2PKH	Pay-to-public-key hash
P2SH	Pay-to-script hash
SPV	Simple Payment Verification
DSKS	Duplicate Signature Key Selection
TLS	Transport Layer Security
IKE	Internet Key Exchange
SSH	Secure Hash

Introduction

Cryptocurrencies such as Bitcoin rely on cryptographic primitives for their guarantees and correct operation. Such primitives typically weaken over time, due to progress in cryptanalysis and advances in the computational power of the attackers. It is therefore prudent to expect that, in time, the cryptographic primitives used by Bitcoin will be partially, if not completely, broken.

In anticipation of such breakage, the Bitcoin community has created a wiki page that contains draft contingency plans. However, these plans are informal and incomplete: no adequate transition mechanism has been built into Bitcoin, and no plans for weakened primitives have been considered. Primitives rarely break abruptly: for hash functions, it is common that first a single collision is found. This is then generalized to multiple collisions, and only later do arbitrary collisions become feasible to compute. In parallel, the complexity of attacks decreases to less than brute-force, and the computational power of attackers increases.

Even if such attacks are years away from being practical, it is crucial to anticipate the impact of broken primitives so that appropriate contingency plans can be put in place. Here the work contributes to filling this gap and provide the first systematic analysis of the impact of broken primitives on Bitcoin. By analyzing the failure of primitive properties, both in isolation and in combination, the range of consequences different breaks have and pinpoint their exact cause are described.

A break in RIPEMD160 can allow an attacker to repudiate payments. SHA256 collisions and second pre-image attacks, as well as selective forgery of Elliptic Curve Digital Signature Algorithm (ECDSA) signatures, all allow an adversary to steal or destroy coins. Finally, SHA256 pre-image attacks have the most severe consequences, as they allow an attacker to take complete control over the Bitcoin system, but only through exploiting the flexibility of the coin base transaction.

The investigations raise concerns about the currently specified migration plans for Bitcoin, being overly conservative in some respects and inadequate in others. To that end, certain suggestions are made regarding future iterations of Bitcoin in response to entirely broken and partially weakened primitives and relate Bitcoins security model to that of other currencies.

Related Works

Identifying how hash collisions break the security of protocols such as TLS, IPSec, and SSH was recently investigated in Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH. More recently, researchers identified vulnerabilities with the cryptographic function used in the IOTA cryptocurrency, which allowed them to create syntactically valid transactions with the same hash, but which pay out different amounts. In terms of the primitives used in Bitcoin, attacks against RIPEMD160 pre-images and collisions as well as SHA256 collisions and pre-images work only for a reduced number of rounds and incrementally improve on brute-force solutions. Certain Elliptic Curve Digital Signature Algorithm (ECDSA) parameters can lead to Duplicate Signature Key Selection, where an adversary can create a different key P′ that validates against a correct signature under a key P. Such research indicates that Bitcoin primitives are indeed under attack, highlighting the need for anticipating their breakage.

More generally, for combining hashes effectively, Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions shows that simultaneous collisions for multiple hash functions are not much harder to find than individual ones. On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak shows that even when the underlying compression functions behave randomly but collisions are easy to generate, finding collisions in the concatenated hash h1(x)||h2(x) and h1(x) XOR h2(x) requires $2^n/2$ queries. However, when the hash functions use the Merkle-Damgrd (MD) construction, there is a generic pre-image attack against the XOR hash with complexity $(2^{5n}/6)$. MD hash functions also behave poorly against pre-image attacks, allowing one to find second pre-images of length 2^{60} for RIPEMD160 in $2^{106} \ll 2^{160}$ time. If an adversary can further find many collisions on an MD construction, he or she can also find pre-images that start with a given prefix.

2.1 Background

Bitcoin is a popular peer-to-peer (P2P) cryptocurrency introduced in 2008 by Satoshi Nakamoto. Fig. 2.1 shows a high-level view of the main component of Bitcoin $theblockchain$ which will guide this section. The blockchain is a public log of all Bitcoin transactions that have occurred, combined together in components called blocks. Transactions use a scripting language that determines the owners of coins, and it is up to miners to verify that only valid transactions occur. To ensure that nobody can change or remove past transactions, miners have to solve a hard computational puzzle, known as a proof of work (PoW). The final component of Bitcoin is its underlying P2P network, which enables distributed communication.

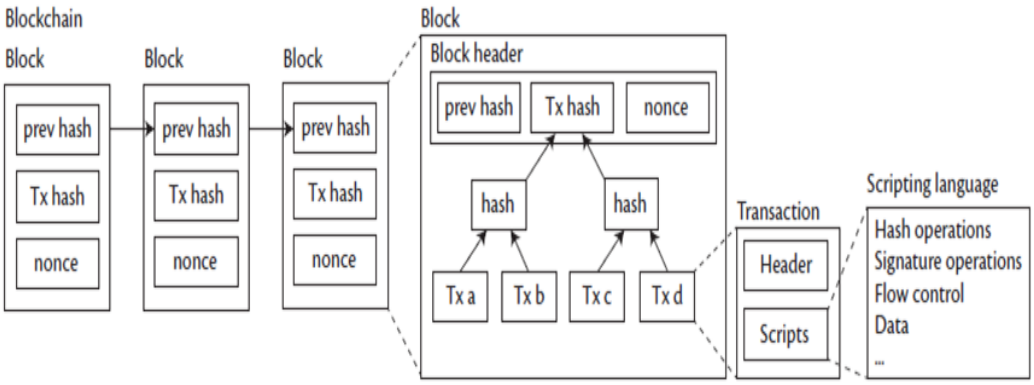


Fig. 2.1: The blockchain data structure

Transactions and Scripts

Bitcoin is an electronic cash system,2 so transactions to transfer coins between users are central to its structure. A transaction is a list of inputsunspent transactions in the blockchainand a list of outputsaddresses to which to transfer the coins, whose unit is a satoshi, equal to 1028 bitcoins or BTCs. To ensure that only the owner can spend his or her coins, each input and output is accompanied by a script. For outputs, this locking script contains the conditions under which the output can be redeemed (scriptPubKey), while for inputs,an unlocking script contains a cryptographic signature (scriptSig) as proof that these conditions have been met. These scripts are sequences of instructions (opcodes) that get executed by miners. To prevent

denial-of-service (DoS) attacks exploiting computationally intensive instructions, most nodes accept only the five standard scripts:

- Public key. The unlocking script must sign the transaction under this key.
- Pay-to-public-key hash (P2PKH). The unlocking script must provide a public key that hashes to the given value (address), and must then sign the transaction.
- Multisignature. An M-of-N (N$\leq$15) multisignature scheme provides N public keys and requires M signatures in the unlocking script
- Pay-to-script hash (P2SH). This script is the hash of a non-P2SH standard transaction. The unlocking script provides the full script hashing to this value and any necessary signatures. This script is typically used to shorten the length of multisignature transactions.
- Data output (OP RETURN). The output cannot be redeemed but can be used to store up to 40 arbitrary bytes, such as human-readable messages.

For a transaction to be valid, it must contain all the required fields, all signatures must be correct, and the scripts must be standard. This is a task that miners undertake for a small fee. In addition, non-standard scripts using different sequences of opcodes can be included in blocks for higher fees.

Mining and Consensus

To ensure that no coin is used more than once, every transaction is made public through a global, append-only ledger called the blockchain, consisting of blocks combining transactions in a Merkle tree. New blocks become a part of the blockchain through a process called mining: miners need to find a value (nonce) such that the hash of a blocks header is less than a given target $h(hdr||nonce), T$. The idea behind this PoW scheme is that the probability of creating the next block is proportional to the miners computational power, and because miners receive transaction fees, they are incentivized to validate transactions and blocks, using the procedure shown below.

Due to the probabilistic nature of mining, the presence of adversaries, and networking delays, miners may disagree on the current state of the blockchain. This is known as a fork. To deal with this issue, thereare hard-coded blocks included in the clients, known as checkpoints, starting from the first block, called the genesis block. In addition, honest (non-adversarial) miners work on the longest blockchain they become aware of, when other nodes announce new blocks and transactions. These temporary forks enable double spending: an adversary can have different transactions in different branches of the fork using the same inputs but different outputs. However, because the probability of deep forks (forks where branches differ in the last N blocks) drops exponentially in N, receivers usually wait for multiple confirmation blocks.

```
input : Bitcoin block
output: valid or invalid
/* Verify block header */
Verify Hash(block header) < target
Verify Merkle hash
Verify Hash(prev block) = prev_hash
/* Verify each transaction input in block */
foreach transaction input in the block do
    Check that referenced output transaction exists and hasn't
     already been spent
    Verify signatures
end
```

Fig. 2.2: Procedure to verify a blocks cryptographic primitives

Network

The last key component is the P2P network for distributed operation. Transactions and blocks are broadcast by nodes to their peers, and then relayed further to flood the network if they meet the relay policies (to prevent DoS attacks). Not every node is a miner or necessarily has access to the full chain: lightweight clients that use Simple Payment Verification (SPV) download only headers and the relevant transactions (with the corresponding Merkle trees).

Over time, the need for extensions or bug fixing motivates protocol changes. Because not all nodes upgrade at the same time, this may introduce forks. If the validation rules in the

upgrade become stricter, then the protocol remains backward compatible, resulting in a soft fork. A hard fork, on the other hand, is not backward compatible and thus requires the entire network to upgrade, as old software would reject new transactions and blocks as invalid.

Proposed System

3.1 Broken Hashing Primitives

Here the cryptographic hash functions in Bitcoin and analysis on the effect of a break in one of the properties of first and second pre-image and collision resistance are defined.

A. Hashing in Bitcoin

In the original Bitcoin paper the concrete primitives used are not specified: there were no addresses but just public keys, and the hash used for mining and the Merkle tree was just referred to as a hash function. The current Bitcoin implementation uses two hash functions.

Main hash

This hash function has an output of 256 bits and requires applying SHA256 twice: $HM(x) = SHA256(SHA256(x))$. It is the hash used for mining (PoW): miners need to find a nonce such that the double SHA256 hash of a block header is less than a target hash. It is also used to hash transactions within a block into a Merkle tree, a structure that summarizes the transactions present within a block. Finally, it is the hash used for transactions signed with a users private key.

Address hash

The second hash function is used as part of the P2PKH and the P2SH scripts. Its output is 160 bits, and it is concretely instantiated as $HA(x) = RIPEMD160(SHA256(x))$.

B. Modeling Hash Breakage

Here the analysis shows how hashes break in terms of their building blocks.

Identifying hashing building blocks

A good cryptographic hash function h(x) should offer three properties:

- Pre-image resistancegiven y, it is hard to find x with h(x)=y
- Second pre-image resistancegiven x1, it is hard to find x2 not equal to x1 with h(x1)=h(x2)
- Collision resistanceit is hard to find distinct x1 not equal to x2 such that h(x1)=h(x2)

We consider attacks against HA and HM abstractly, so that our arguments can be extended for any future version that uses the same structure. Table 1 contains a summary of the results.

C.Main Hash

Here the main hash HM is analyzed, which is used for mining, in Merkle trees, and with signatures. All three use cases are discussed separately.

Mining

The first investigation is the pre-image attacks against the block headers under two different attack scenarios,before turning to collision and second pre-image attacks.

Attack 1: Pre-image against fixed Merkle root. The probability that an adversary with access to a pre-image oracle can break mining is negligible. Miners search for block headers whose n-bit hash is below a target, which we assume starts with d zeros. This assumption introduces only up to 1 bit of extra work, as there is always a unique d with $T \leq 2^d < 2T$ for any target T.

If the adversary controls b≤n bits of the input, there are 2^b possible inputs to the hash function. These need to map to one of the 2^{nd} values in the range $[0^n, 0^d1^{nd}]$ and will be uniformly distributed across 2^n values. The adversary can only query the pre-image oracle for specific target hashes. Because there are 2^{bd} b-bit pre-images distributed across the 2^{nd} values, the probability that a given hash in $[0^n, 0^d1^{nd}]$ has a b-bit pre-image is: $P[correctpre-image] = \left(2^{bd}\right) / \left(2^{nd}\right) = 2^{bn}$. This probability does not depend on d, as one might expect. This is because, by increasing d to reduce the number of valid hashes, the adversary also reduces the expected number of b-bit pre-images. Assuming the adversary is allowed 2a queries to the oracle, the probability of breaking mining becomes $P[success] = 2^a.2^{bn} = 2^a1^{bn}$.

To calculate b, we explore all fields in the block header. The version number (nVersion), the hashes of the previous block header (hashPrevBlock), and the hashes of the current Merkle root hash (hashMerkleRoot) are fixed. However, the adversary has partial control over the remaining fields in the header. For the timestamp field (nTime), the value can be within 7,200 seconds of the current median timestamp, giving the adversary approximately 13 bits of freedom. Moreover, the adversary has complete control over the 32 bits of the nonce (nNonce). The nBits field 0xAABBCCDD describes the target difficulty as 0xBBCCDD . 256^{0xAA3} , with the protocol checking only that the produced number is at most the target value given by the consensus. At the time of writing, the target value is 0x1743eca9, granting the adversary approximately 27 bits of freedom. Together the fields give $b = 72$. With $n = 256$, and allowing 2^{80} calls to the oracle, the probability of success is only $2^{80+72-256} = 2^{-104}$, which is negligible.

Attack 2: Pre-image against variable Merkle root. By varying the Merkle root, an adversary can break mining, although per the discussion of Attack 1, this cannot be achieved by simply reordering or excluding transactions. Instead, the adversary must work backward by querying the oracle for a target Merkle hash and repeatedly querying the oracle to reconstruct the entire Merkle tree. This would normally fail, as the transactions generated would not be valid due to incorrect signatures (although transaction malleability can allow the same signature to be valid for two different transactions3), but Bitcoin does not enforce a minimum number of transactions in a block. Hence, miners can mine blocks with just the coinbase transaction,

which generates new coins and which has a variable-length input of up to 100 bytes that is controlled by miners. A malicious miner with access to the pre-image oracle can then,

- Pick an arbitrary target T and get a pre-image for $HM(a||x||b) = T$, where the desired x is the hashMerkleRoot field and a,b are the remaining fields in a block header. Because the root is 256 bits, there is a pre-image with high probability, but if not, repeat with some other random target T'
- Pick a length l for the script, and fix all other fields for the coinbase transaction. Solve $HM(a'||y||b') = x$ where a', b' are the remaining fields for the coinbase transaction. Because the number of free bytes is up to 100, there is an l-bit pre-image y with high probability. The miner then generates a coinbase transaction using a', y, b' and combines it into a block using a,b. This block will have a hash of T as desired.

Collisions, second pre-images: Collisions and second pre-images are useful for mining only if the pre-images start with d zeros. Assuming the pre-images contain valid transactions and signatures, a miner can fork the chain, but this only occurs with negligible probability.

Merkle trees: Breaks of the main hash can also be used to attack Bitcoin through the Merkle trees used to store transactions in blocks.

Attacking existing blocks

A similar argument as for mining (Attack 1) shows that an adversary cannot find a valid second pre-image of an entire block except with negligible probability. Pre-images do not give the adversary new information, as they already accompany the hash value. Collisions are also not useful, as both values are attacker controlled and cannot alter existing blocks.

Attacking new blocks

For new blocks and transactions, an adversary with sufficient network control can use a collision or second pre-image to split the network, reject both blocks, or reverse transactions, thus enabling double spending. Pre-images are again not useful, as they always accompany the hashed value.

Main hash usage in signatures: In Bitcoin, signatures are over messages hashed with HM. Therefore, a second pre-image attack or a collision on HM can be used to destroy and possibly steal coins: an adversary can ask for a signature on an innocuous transaction (for example, pay 1 satoshi from address X to address Y) but transmit a malicious one instead (for example, pay 100 BTC from address X to address Z) because the adversary controls enough bytes to guarantee success with high probability. Note that for this attack to succeed, the adversary must still specify the same unspent transaction X belonging to the victim for the signature to be valid, but can alter the amount (bounded by the total number of bitcoins present in address X) and the destination.

Address Hash

The address hash is used in two contexts. First, in P2PKH scripts a Bitcoin address is essentially $y = HA(p) = RIPEMD160(SHA256(p))$ where p is the public key (together with a checksum). Payments to addresses use only the hashed value y, but transactions from addresses require the full public key p and the signature on the transaction. The second use is in P2SH scripts. A P2SH is $y = HA(s)$, where s is a standard script, typically a multisignature transaction. Payments to a P2SH script do not reveal the pre-image, but transactions spending the coins require it and the signatures of the corresponding parties. We discuss them jointly, since the only difference between a P2PKH and a P2SH in this context is the number of required signatures.

Pre-image

For previously spent outputs or for reused addresses, HA is already accompanied by its pre-image. A pre-image thus can only reveal the public key(s) for unspent outputs. This has minimal privacy consequences because public keys are not tied to real identities, but it could enable an offline attack on the key. Assuming that the key was not chosen with bad randomness and there is no weakness in the signature scheme, the probability of success is still negligible.

Second pre-image

A second pre-image gives the adversary access to a different public key or script with the same hash. However, because the adversary does not control the corresponding private key,

he or she cannot use this to change existing transactions or create new ones. This is because pre-images (whether a key or a script) are revealed and verified only when spent in transactions.

Collision

Collisions are similar, although in this case, both public keys are under the adversarys control and again the adversary does not have access to the private keys. In both scenarios, there is a question of non-repudiation external to the protocol itself: by presenting a second pre-image of a key used to sign a transaction, a user/adversary can claim that his or her coins were stolen.

Breakage	Address hash (H_A)	Main hash (H_M)
Collision	Repudiate payment	Steal and destroy coin
Second pre-image	Repudiate payment	Double spend and steal coins
Pre-image	Uncover address	Complete failure of th blockchain

Fig. 3.1: Summary of the effects on Bitcoin for different types of hash breakages

3.2 Broken Signature Primitives

Here the use of digital signatures in Bitcoin is described and analyze how a break in their unforgeability, integrity, or non-repudiation impacts Bitcoin. Fig. 3.2 summarizes our results

A. Digital Signatures in Bitcoin

Bitcoins digital signature scheme is ECDSA with the secp256k1 parameters, and is used to sign the main hash HM of transactions.

Modeling Signature Breakage Variants

The security of digital signature schemes is usually discussed in terms of three properties, which defined as follows:

- Unforgeability no one can sign a message m that validates against a public key p without access to the secret key s;
- Integritya valid signature ms does not validate against any $m'm$;
- Non-repudiationa valid signature ms does not validate against any public key $p'p$

where there is an implicit except with negligible probability, due to hashing.

These properties are linked, and a breakage in one usually implies a breakage in the others. In addition, they are often discussed in a much more abstract way: non-repudiation refers to the property that the signature proves to all parties the origin of the signature, but in this case, a way is introduced that is more akin to Duplicate Signature Key Selection (DSKS) attacks.

B. Broken Signature Scheme Effects

A break in each of the properties are analyzed separately, starting with the last two, as neither of them can lead to an attack on their own.

Integrity

For a break in the integrity of the signature scheme to be useful in Bitcoin, a signature of HM(m) must also be valid for HM(m'). This involves HM in a non-trivial way, so we discuss this further later, but note that transaction malleability can cause the issuer of a transaction to think that his or her payment was not confirmed.

Non-repudiation

For non-repudiation, note that for transactions, even if a signature verifies under a different key, the address hashes of the two public keys must match. A break thus involves HA.

Breakage	Effect
Selective forgery	Steal coins from public key
Integrity break	Claim payment not received
Repudiation	-

Fig. 3.2: Effects of a broken signature scheme

Unforgeability

When it comes to unforgeability,various types of breaks can be found :5 total break to recover the private key, universal forgery to forge signatures for all messages, selective forgery to forge a signature on a message of the adversarys choice, and existential forgery to produce a valid signature that is not already known to the adversary.

Because the message to be signed must be the hash of a valid transaction, an existential forgery is not sufficient because the probability that it corresponds to a valid message is negligible. Selective forgery, on the other hand, can be used to drain a victims wallet. From this perspective, selective forgery and a total break have the same effect. However,the type of breakage influences how to upgrade to a new system. It is worth noting that an adversary does not necessarily have access to a users public key, because addresses that have not been reused are protected by the address hash HA.

3.3 Multi-Breakage

Here how combinations of breakages in different primitives can impact Bitcoin is analyzed. Because HA and HM are not used together, we consider only a break in the signature algorithm in combination with a break in one of the two hashes. The results are summarized in Fig. 3.3 .

A. Address Hash and Signature Scheme

A combined break of the address hash and the signature scheme allows an adversary to steal coins, and repudiate or replace transactions.

Signature forgery

A combination of a selective forgery with a first or second pre-image break of the address hash can be used to steal all coins that are unspent. Generating two public keys p,p′ with $HA(p) = HA(p')$ (collision) whose signatures the adversary can forge does not have a direct impact, because the adversary controls both addresses. However, it appears as if two different users are attempting to use the same coin, thus raising a question of repudiation.

Signature integrity

As the messages signed for transactions do not involve HA, this combination does not increase the adversarys power

Signature repudiation

A pre-image attack on HA is not useful as the public key is already known. For a second pre-image, assume that given a message m (the hash of a transaction) and a public key p, an oracle returns p′ such that HA(p)= HA(p′) and the signature of m under p also validates against p′ . Because the same signature validates for both keys, an adversary can replace p by p′ in the unlocking script. Although this does not give the adversary immediate monetary gain, a transaction in the blockchain has been partially replaced. For collisions, assume that given a message m, an oracle returns two public keys p,p′ such that HA(p)= HA(p′) and the signature of m under p validates under p′ .If the adversary does not have access to the private keys, he or she cannot sign the transaction. Otherwise, the effect is identical to the second pre-image case,

where the adversary can replace part of a transaction in the blockchain.

Hash property	Signature property		
	Selective forgery	Integrity break	Repudiation
Address hash (H_A)			
Collision	Repudiate transaction	-	Change existing payment*
Second pre-image	Steal all coins	-	Change existing payment
Pre-image	Steal all coins	-	-
Main hash (H_M)			
Collision	Steal coins	Steal coins*	-
Second pre-image	Steal coins	Double spend*	-
Pre-image	-	-	-

Fig. 3.3: Multi-breakage effects:combining broken hashes and signatures

B. Main Hash and Signature Scheme

A combined break of the main hash and the signature scheme allows an adversary to steal or double spend coins.

Signature forgery

For mining, a pre-image attack is useful when working backward from a fixed target to get a pre-image for the Merkle root and turn it into a tree of transactions. The problem that is identified earlier was that there is only negligible probability that the transactions refer to valid, unspent outputs, so a forgery does not solve this issue. Finally, for transactions, collisions and second pre-images on their own can be used to destroy or steal coins. If adversaries can also forge signatures, they are guaranteed to be able to steal coins no matter what address they went to, as long as it is not protected by the address hash.

Signature integrity

A collision or a second pre-image attack trivially breaks the integrity of the scheme, as messages are always hashed, and reduces to the case about main hash usage in signatures.

Thus modify the definitions slightly to consider a joint break in the two algorithms.

A collision integrity oracle given a public key p produces m,m′ such that the signature of HM(m) is also valid for HM(m′). The adversary can ask for a signature on an innocent transaction but transmit the malicious one with the still valid signature. Unlike in the regular collision case, the two hashes HM(m) and HM(m′) are different. Hence, the adversary cannot just replace the transaction in the block, but he or she can opt never to transmit the innocent one instead.

A second pre-image integrity oracle given a public key p and a message m produces m′ such that the signature of HM(m) is also valid for HM(m′). This case also resembles the break on just HM, but again, because the hashes are not equal, the adversary cannot simply replace an existing transaction, unless it has not yet been confirmed in a block. This can split the network and destroy coins.

Signature repudiation

The non-repudiation property of the signature scheme necessarily involves a break of HA. This combination therefore does not increase the adversarys power.

3.4 Current Bitcoin Implementation

Here the current Bitcoin implementation, in the context of its choice of primitives, non-standard scripts, and contingency plans, using observations from the previous sections are revisited.

A. Current Cryptographic Primitives

In the current implementation of Bitcoin, HA(x)= RIPEMD160(SHA256(x)), and HM(x)= SHA256 (SHA256(x)). Because there are no critical breaks for HA, a break in RIPEMD160 is not cause for concern. Moreover, because HM uses only SHA256, an attack against SHA256 is equivalent to an attack against HM.Thus the effect of concrete primitive breakage are sum-

marized as shown in Fig. 3.4.

Breakage	Effect
SHA256	
Collision	Steal and destroy coins
Second pre-image	Double spend and steal coins
Pre-image	Complete failure
RIPEMD160	
Any of the above	Repudiate payments
ECDSA	
Selective forgery	Steal coins
Integrity break	Claim payment not received
Repudiation	-

Fig. 3.4: Effects of concrete primitive breakage on the current version of Bitcoin

B. Non-Standard Scripts

Again, the Bitcoin scripting language extends beyond the five standard types of transactions. In part due to the higher fees associated with them, non-standard transactions represent less than 0.02 percent of all bitcoins in circulation (https://p2sh.info/dashboard /db/non-standard-outputs-statistics) but are more versatile and can include opcodes for calculating SHA1, SHA256, and RIPEMD160 hashes. By using these opcodes, one can create challenges involving the primitives, so that, for instance, a user needs to create a collision to redeem a certain coin.

One set of such challenges was created by an early Bitcoin developer asking for collisions on individual and combined primitives (https://bitcointalk.org/index .php?topic5293382.0). The challenge for a SHA1 collision was claimed using the collision found by Stevens and colleagues6 for a bounty of 2.48 BTC, the equivalent of approximately 2,800 dollars at the time.

Although the USD/BTC exchange rate fell temporarily as a result of this news, it quickly recovered, because SHA1 is not an integral part of the Bitcoin protocol. However, this collision highlights the need to anticipate the breakage of primitives, because non-standard transactions allow collision and pre-image attacks to be used even when the core protocol is not yet broken. However, as we explain, the existing contingency plans are not sufficient.

C. Existing Contingency Plans

A break of the primitives has interested the community from the early days of Bitcoin. Informal recommendations by Satoshi in forums evolved into a wiki page that describes contingency plans for catastrophic failure[s].1 Such a failure for primitives is defined in terms of an adversary that can defeat the algorithm with a few days of work,1 and the focus is on notifying users and protecting the OPCHECKSIG operation to prevent people from stealing coins.

Concretely, for a severe, 0-day failure of SHA-256,1 the plans propose switching to a new hashing algorithm H' and hard-coding known public keys with unspent outputs as well as the Merkle root of the blockchain under H' . For a broken signature scheme, if the attacker cannot recover the private key and there is a drop-in replacement using the same key pair, the plan is to simply switch over to the new algorithm. Otherwise, the new version of Bitcoin should automatically send old transactions somewhere else using the new algorithm.

D. Potential Migration Pitfalls

The contingency plans suggest that code for all of this should be prepared, but no such mechanism currently exists. Moreover, no plans are in place for a break in RIPEMD160. Because sudden breaks are unlikely, neither is cause for immediate concern, but should be included in future plans.

Broken SHA256

It is clear from the analysis that new transactions should not use a broken hash. However, existing historical transactions and blocks cannot be altered, except in a majority

mining attack. Thus, hard-coding public keys and rehashing the entire blockchain are more prudent than necessary. It should be noted that a sudden break necessitates a hard fork for Bitcoin.

Broken ECDSA

For a broken ECDSA, a transition is indeed easy if there is a drop-in replacement and the private key is safe. Otherwise, a gradual transition scheme is necessary as users will need to manually switch over to a new key pair.

E. Recommendations

Here recommendations to more properly anticipate primitive breakages are made. Recognizing that there are financial considerations in addition to the technical ones, we do not propose a full upgrade mechanism but merely make suggestions to the Bitcoin developers and community.

First of all, the analysis reinforces the idea that users should not reuse addresses, not just for privacy reasons but also because this protects against some types of primitive breakage. For instance, if the signature scheme is broken, addresses are still protected by the hash.

The plans for a sudden breakage should address when to freeze the blockchain and whether to roll back transactions in the case of a sudden break. Moreover, the centralized approach of hard-coding well-known keys is perhaps not entirely in line with Bitcoins decentralized philosophy and can lead to lost coins. If keys are to be hard-coded, there is a tradeoff between complexity and risking making coins unspendable: developers must decide whether the migration would occur at once or whether new key pairs should be distributed periodically. An alternative and perhaps better approach would be to use zero-knowledge proofs to tie the old address still protected by its hash to the new public key.

Given that sudden breaks are unlikely, there is a need for a separate plan for weakened primitives. The analysis the recommendations are the following:

- Introduce a minimum number of transactions per block to increase the difficulty of performing the pre-image attack against the mining header target (PoW) using the coinbase

transaction.

- To migrate from old addresses, whether due to a weakened hash or signature scheme, introduce new address types using stronger hashing and signature schemes. This can be achieved with a soft fork by making transactions appear to old clients as pay-to-anybody, akin to how P2SH was introduced.

- Instead of using nested hashes for HA, HM, combine primitives in a way that increases defense in depth.

- Given that HM has multiple use cases, consider whether each of its functions should have a different instantiation, whether through distinct primitives, by prepending different values, or by using an HMAC with different keys.

- Consider a hard fork in response to a weakened HM, with redesigned headers and transactions and without any use of the old primitives.

A soft fork is insufficient for properly upgrading a weakened hash function HM= H1 to the stronger H2, because HM forms the core of the PoW scheme. Specifically, because any changes must be backward compatible, the old validation rules must still apply, so for every new block, H1(hdr) , T, where the target T is still calculated by the same algorithm. New blocks would also need to satisfy some additional constraint H2(hdr′) , T′ , where the target T′ is calculated independently and hdr′ is the block header, possibly excluding some fields. As a result, new clients would have to solve two PoW computational puzzles. Although every instance of H1 (transaction, Merkle root, and so on) could be accompanied by an instance of H2, blocks and transactions are fundamentally identified by their H1 hash, which an attacker could exploit. There are also questions of incentives and whether new iterations of Bitcoin would still use a PoW scheme, but this is left as future work.

F. Other Cryptocurrencies

Bitcoin is the largest cryptocurrency by market capitalization and adoption, but both derivatives (altcoins) and completely different designs have spread. Although discussing them all is out of scope, we identify a few common threads among those that use PoW schemes.

The first class of altcoins consists of those that are forks of Bitcoin with additional types of transactions supported. For these currencies, our analysis applies verbatim but needs to be extended to these new types of operations. As an example, Namecoin introduces transactions for registering and updating .bit domains. The name new $< name >$ command creates a transaction whose outputs include $HA(r||name)$, where r is a nonce. This acts as a pre-order for the domain name .bit, which is then updated by its owner after the transaction has been in 12 confirmed blocks. This pre-order commitment phase is necessary to ensure no one can steal the name of the domain, which is fully registered in a subsequent namefirstupdate command that reveals the name, the nonce, and the DNS configuration values, signing the transaction with the users key.

Any updates for a domain require signatures, so any novel exploit would involve the namenew and namefirstupdate commands. Unlike Bitcoin, however, a pre-image attack on HA allows recovering a random nonce and domain name. Namecoin rules do not seem to preclude namenew transactions with the same hash, but if the pre-image recovers the original (r,name) pair, an attacker can pre-register the domain by publishing his or her own namenew transaction before the original users transaction is confirmed. In practice, however, this attack would require pre-image oracles for both hash functions used in HA (SHA256, RIPEMD160), implying that there is a pre-image oracle for HM as well, making the entire currency insecure.

The second set of cryptocurrencies consists of those that separate the roles of HM and replace it by two different hash functions, say HI for integrity checking and HP for the PoW. If HI breaks, one can still steal coins by creating collisions in transactions, which are hashed before they are signed. However, because HI is not used for PoW, an attack on HI cannot directly be used to exploit mining. More importantly, even a full pre-image oracle on HP is not sufficient for a breakdown of the currency. This is because the probability of finding a valid pre-image for a fixed target is negligible, as the adversary does not control sufficiently many header

bits. Instead, as also identified earlier, an adversary needs to be able to alter the Merkle root at will, which is protected by the different HI, requiring both to be exploited for a successful attack. Example currencies employing this scheme include Litecoin and Dogecoin, where HI= SHA256(SHA256), and HP is the scrypt key derivation function. Ethereum is similar, with HI using Keccak-256 and HP using Ethash.

Primecoin has a different PoW mechanism. It is a fork of Bitcoin using the same structures (and only the single HM), but its PoW requires finding long chains of prime numbers pi that satisfy p$i + 1$ =2 p$i + 1$ or p$i + 1$ =2 p$i - 1$, with $p0 = k.HM(header) + 1$ or $p0 = k.HM(header) - 1$ for some k (technically, Cunningham chains of first or second kind, or bi-twin chains). An adversary with access to a pre-image oracle can attack this scheme by reusing existing chains: if (pi) is a valid chain for header h and the difficulty has not changed, then (pi) is also a valid chain for h/q for any factor q of h, and for h.r, where r is a small factor of k (to ensure that h.r remains a valid hash). Note that once more, this exploit depends on using the pre-image oracle twice: once for the overall header and once for the coinbase transaction, showing that it is necessary to increase the minimum number of transactions per block.

Finally, there are currencies that use a hybrid PoW and proof-of-stake or proof-of-burn scheme. For instance, Peercoin is a fork of Bitcoin that uses the same PoW and structure as Bitcoin, but also allows minting coins based on their age for a proof-of-stake approach. As a result, the attacks identified in this article still apply, but the attack surface increases to include the new transactions and needs to be analyzed separately.

Conclusion

Here the first systematic analysis of the effect of broken primitives on Bitcoin are explained. The analysis reveals that some breakages cause serious problems, whereas others are inconsequential. The main vectors of attack involve collisions on the double SHA256 hash or attacking the signature scheme, which directly enables coin stealing. In contrast, a break of the hash used in addresses has minimal impact, because addresses do not meaningfully protect the privacy of a user. The analysis has also uncovered more subtle attacks. For example,the existence of another public key with the same hash as an address in the blockchain enables parties to claim that they did not make a payment. Such attacks show that an attack on a cryptographic primitive can have social rather than technical implications. In most cases, the failure of a single property in one cryptographic primitive is as bad as multiple failures in several primitives at once. For future versions of Bitcoin, the recommendations include various redundancies, such as properly combined hash functions, and a minimum number of transactions per block. Bitcoins migration plans are currently under specified and offer at best an incomplete solution if primitives get broken. We offer some initial guidelines for making the cryptocurrency more robust not only for a sudden break but also in response to weakened primitives. However, future discussions should directly involve the Bitcoin developers and community to propose plans that would be in line with their expectations.

References

[1] ”Contingency Plans,” Bitcoin Wiki, 2015; https://en .bitcoin.it/wiki/Contingency plans.

[2] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008; https://bitcoin.org/bitcoin.pdf.

[3] R. Ali and A.K. Pal, A secure and robust three factor authentication scheme using RSA cryptosystem, Int. J. Bus. Data Commun. Netw., vol. 13, no. 1, pp. 74-84, 2017.

[4] C. Decker and R. Wattenhofer, Bitcoin Transaction Malleability and MtGox, European Symposium on Research in Computer Security (ESORICS), 2014.

[5] S. Goldwasser, S. Micali, and R.L. Rivest, A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks, SIAM Journal on Computing (SICOMP), 1988.

[6] M. Stevens et al., The First Collision for Full SHA-1, Annual International Cryptology Conference (CRYPTO), 2017

[7] K. Bhargavan and G. Leurent, Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH, Network and Distributed System Security Symposium (NDSS), 2016

[8] E. Heilman et al., IOTA Vulnerability Report: Cryptanalysis of the Curl Hash Function Enabling Practical Signature Forgery Attacks on the IOTA Cryptocurrency, 7 Sept. 2017; https://github.com/mit-dci/tangled-curl/blob /master/vuln-iota.md.

[9] C. Ohtahara, Y. Sasaki, and T. Shimoyama, Preimage Attacks on Step-Reduced RIPEMD-128 and RIPEMD-160, International Conference on Information Security and Cryptology (Inscrypt), 2010

[10] F. Mendel et al., Improved Cryptanalysis of Reduced RIPEMD-160, International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), 2013

[11] F. Mendel, T. Nad, and M. Schlffer, Improving Local Collisions: New Attacks on Reduced SHA-256, Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), 2013

[12] D. Khovratovich, C. Rechberger, and A. Savelieva, Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family, International Workshop on Fast Software Encryption (FSE), 2012