Time-release Protocol from Bitcoin and Witness Encryption for SAT

Jia Liu
School of Computer Science
University of Birmingham
liujw@cs.bham.ac.uk

Flavio Garcia
School of Computer Science
University of Birmingham
f.garcia@bham.ac.uk

Mark Ryan
School of Computer Science
University of Birmingham
m.d.ryan@cs.bham.ac.uk

ABSTRACT

We propose a new time-release protocol based on the bitcoin protocol and witness encryption. We derive a "public key" from the bitcoin block chain for encryption. The decryption key are the unpredictable information in the future blocks (e.g., transactions, nonces) that will be computed by the bitcoin network. We build this protocol by witness encryption and encrypt with the bitcoin proof-of-work constraints. The novelty of our protocol is that the decryption key will be automatically and publicly available in the bitcoin block chain when the time is due.

Witness encryption was originally proposed by Garg, Gentry, Sahai and Waters. It provides a means to encrypt to an instance, x, of an NP language and to decrypt by a witness w that x is in the language. Encoding CNF-SAT in the existing witness encryption schemes generate $\mathsf{poly}(n \cdot k)$ group elements in the ciphertext where n is the number of variables and k is the number of clauses of the CNF formula. We design a new witness encryption for CNF-SAT which achieves ciphertext size of $2n + 2k$ group elements. Our witness encryption is based on an intuitive reduction from SAT to Subset-Sum problem. Our scheme uses the framework of multilinear maps, but it is independent of the implementation details of multilinear maps.

Keywords

time-release crypto, bitcoin mining, proof-of-work, witness encryption, SAT, Subset-Sum, multilinear maps

1. INTRODUCTION

Security researchers have often received legal threats from vendors and government agencies seeking to stop publishing vulnerability information or "proof of concept" code demonstrating the security flaws. A long list of legal threats against security researchers can be found in [2]. On one hand, discovering security flaws and releasing a public advisory on vulnerabilities can help vendors repair their products and inform and protect the customers. On the other hand, vendors fear negative publicity challenges of the security of their products. Also the vulnerability information can give attackers information to exploit a security hole and cause harm.

We picture a new (conference) publication system based on time-release protocol to mitigate the above issues. Image Alice wants to publish a paper about security flaws of some product on Usenix Security, but the vendor of the product tries to inhibit her. Instead, Alice can publish an encrypted version of her paper by using a time-release encryption scheme. Her paper can only be decrypted after a certain period of delay, e.g., 90-day. This delay is the notice period which gives the vendor time to fix their products. When the 90-day deadline hits, the decryption key will be available to the public and everyone can decrypt. During this waiting period, Alice cannot change her paper and the vendor cannot change their mind to thwart the recovery of the decryption key.

1.1 Time-release crypto

The concept of time-release crypto was first suggested by May [25] in 1993 and further studied by Rivest, Shamir and Wagner [23]. The goal is to "send information into the future". This problem has been found to have many real-world applications such as electronic auctions, scheduled payment methods, sealed-bid auctions, lotteries. Two essential approaches have been investigated to solve the problem: the computational complexity approach [23, 4, 6] and trusted third party [13, 23, 5, 8, 22].

Our proposed time-release protocol is based on computational complexity. The nolvety of our protocol is that we integrate the protocol with the existing decentralised bitcoin protocol. As a result, the decryptor in our protocol does not have to invest the computational effort to decrypt. The bitcoin network will do the computational work instead. In the previous time-release crypto, the actual decryption time depends on the moment at which the decryption starts and the computational power the decryptor owns. The decryption will be delayed if the decryptor starts late. In comparison, in our protocol, when the time is up, the decryption key will be automatically and publically available in the bitcoin block chain. Moreover, we do not have to modify the existing bitcoin protocol. The integration with bitcoin protocol is seamless and the bitcoin protocol functions like a public

time clock.

The basic idea of our time-release protocol is to extract a "public key" from bitcoin block chain for encryption, and the decryption key will be the information in bitcoin blocks (e.g., transactions, nonces, hashes) that will only be determined in future by bitcoin network. Clearly, the difficulty of building such a protocol is that the decryption key is unpredictable. However, among all of those unpredictable information, there is one thing that is actually predictable and can be used as a public key for encryption. That is the bitcoin proof-of-work constraints:

$$\mathsf{SHA256}\left(\mathsf{SHA256}\left(block\_header\right)\right) \leq target$$

The block chain in the bitcoin protocol is an ordered, back-linked list of blocks of transactions. Each block is identified by a hash, generated using the SHA256 cryptographic hash algorithm on the header of the block. For a new block to be accepted by bitcoin network as the next head of block chain, its block header must satisfy the above constraint. Hence the "public key" for our time-release encryption will be the proof-of-work constraints, and witness encryption [16] can allow us to encrypt with a constraint and decrypt with any solution of the constraint.

A new block in the bitcoin network is created every 10 minutes on average and this can be controlled by adjusting the value of the *target*. The smaller the *target* is, the longer it takes to generate a valid block. We specify a recent block as a starting point in the "public key" for encryption and we can customise the time-delay we want in the encryption by choosing the right number of blocks in the proof-of-work constraints, for example 5 blocks for 50 minutes delay from the specified starting point. Similar to all the previous computational complexity based time-release crypto, the accuracy of the time-delay is hard to achieve. The users do not have to wait for the bitcoin network to compute the values that satisfy the hash constraint; they can always compute by themselves if they have enough computing power. However, the bitcoin network possesses a huge computing power (currently around 30,000 Tera hashes per second) and it is unlikely for any individual to acquire such power alone.

1.2 Witness encryption

Witness encryption, originally proposed by Garg, Gentry, Sahai and Waters in [16], as a means to encrypt to an instance, x, of an NP language and decrypt by a witness w that x is in the language. The GGSW construction for witness encryption is based on the Exact-Cover problem and multilinear maps [7]. The efficiency of encoding CNF-SAT depends on the reduction which is unspecified in [16]. As far as we know, the best reductions from CNF-SAT to Exact-Cover is CNF-SAT $\rightarrow$ 3-CNF-SAT $\rightarrow$ Exact-Cover (The details of the second reduction can be found in Appendix C). However, for a CNF formula of n variables and k clauses, the reduction from CNF-SAT to 3-CNF-SAT increases the number of variables and clauses by the size of the original CNF formula, that is the total number of literals occurred in the formula which is $O(n \cdot k)$ in the worst case. Suppose the resulting 3-CNF has n' variables and k' clauses, then the reduction to Exact-Cover generates $2n' + 7k'$ sets. Hence even if it is on 3-CNF, our ciphertext is smaller and only consists of $2n' + 2k'$ elements, and 3-CNF usually has a large number k' for clauses.

The security of GGSW construction [16] is based on instance dependent family of assumptions that they called the "Decision Graded Encoding No-Exact-Cover Problem." To improve the security assurances, Gentry, Lewko and Waters proposes a proof framework in [17] for proving witness encryption schemes secure under instance independent assumptions. The security analysis of their constructions of witness encryption is based on multilinear subgroup decision assumption and multilinear subgroup elimination assumption which are affected by the recent attacks on multilinear maps [9, 18]. Moreover, to compensate the security factor loss caused by their proof technique, the security parameters must be set quite large. There are three instantiations of witness encryption for CNF formulas in [17]. Two of them are specific to the composite order multilinear groups. The conversion from composite-order construction to prime-order multilinear groups (or more generally, groups of arbitrary order) results in a ciphertext of $n \cdot (4n+1)^2 \cdot (n+k)^2$ group elements.

Program obfuscation is the task of making code "unintelligible", so that the obfuscated code reveals nothing about the implementation beyond its functionality. Garg, Gentry, Halevi, Raykova, Sahai, and Waters propose the first general purpose indistinguishability obfuscator ($i\mathcal{O}$) [15] based on multilinear maps [7]. Conceptually program obfuscation generalises witness encryption, but witness encryption is in principle easier to achieve in terms of security and efficiency. This is because program obfuscation must hide the details of the program while witness encryption does not have to. The obfuscation size is typically of $\mathsf{poly}\left(s, 2^d, n\right)$ group elements for circuits of size s, depth d and input length n [15, 3, 28].

1.3 Contributions

Our first contribution is to design a new time-release protocol based on bitcoin. We derive a "public key" from the bitcoin block chain for encryption. The decryption key will be the unpredictable values (e.g., transactions, nonces) that will be computed in future by bitcoin network. We build this protocol by witness encryption and encrypt with the bitcoin proof-of-work constraints..

Our second contribution is to propose a new witness encryption for CNF-SAT which achieves ciphertext size of $2n + 2k$ group elements, where n is the number of variables and k is the number of clauses of the CNF formula. Our witness encryption is based on an intuitive reduction of SAT to Subset-Sum problem and the framework of multilinear maps. In fact, the reduction itself is in the size of the formula which is $n \cdot k$ in the worst case, but we can compress and represent it by just $2n + 2k$ elements in encryption.

1.4 Outline

The rest of this paper proceeds as follows. In Section 2 we provide a brief overview of the Bitcoin protocol. In Section 3, we propose our new time-release protocol. In Section 4, we propose our new witness encryption. In Section 5, we discuss the instantiation of our witness encryption and the

Field	Size	Description
Version	4 bytes	Block version number
hashPrevBlock	32 bytes	Reference to the hash of the previous block header
hashMerkleRoot	32 bytes	The root of the merkle tree of all of the transactions in the block
Timestamp	4 bytes	The approximate creation time of this block
Bits	4 bytes	The proof-of-work difficulty target for this block
Nonce	4 bytes	A counter used for the proof-of-work

Figure 1: The structure of the block header

plausibility of the design of our time-release protocol. The paper concludes in Section 6.

2. OVERVIEW OF BITCOIN PROTOCOL

In this section we provide a short overview of the Bitcoin protocol [24].

Bitcoin is a peer-to-peer electronic cash which allows users to directly transact without going through any central authority or financial institution. Everyone can take part in managing transactions and issuing bitcoins. Transactions are broadcast to and verified by bitcoin network nodes and recorded in a public distributed ledger called the *block chain*.

The block chain is an ordered, back-linked list of blocks of transactions. Each block is identified by a hash, generated using the SHA256 cryptographic hash algorithm on the header of the block. The data structure of block header is given in Figure 1. Each block references a previous block through the "hashPreBlock" field in the block header, as shown in Figure 2. The fields of difficulty, timestamp, and nonce in Figure 1 are related to the mining competition. The merkle tree root is a data structure used to efficiently summarize all the transactions in the block.

To generate the next block, bitcoin peers compete to solve a proof-of-work based on identifying specific SHA256 preimages, specifically to find a nonce, as part of the bitcoin block header, hashes below a certain value:

$$\mathsf{SHA256}\left(\mathsf{SHA256}\left(block\_header\right)\right) \leq 0^{\ell} \| 1^{256-\ell} \qquad (1)$$

When mining bitcoin, the proof-of-work algorithm repeatedly hashes the block header while incrementing the counter Nonce. Whenever Nonce overflows, an extra nonce portion of the generation transaction is incremented, which changes the Merkle root. The value ℓ is the number of leading zeroes which represents the difficulty of bitcoin mining. The bigger ℓ is, the longer the bitcoin mining takes. The difficulty is selected by a periodic network vote to ensure that on average a block is created every 10 minutes. When a peer generates a valid solution, it broadcasts the new block to all nodes in the network. If the block is valid, then the new block is accepted as the head of the block chain.

3. TIME-RELEASE PROTOCOL FROM BITCOIN

We design a new time-release protocol by using bitcoin block chain as a time clock. We derive a "public key" from bitcoin block chain for encryption, and the decryption key will be the unpredictable information in bitcoin blocks (e.g., transactions, nonces, hashes) that will only be determined in future by bitcoin network. We can specify how much time-delay we want in the encryption. When the time is due, the decryption key will be automatically and publicly available in the bitcoin block chain. Bitcoin network is decentralised and there is no trusted third party involved in our protocol.

The future values of blocks are unpredictable since we cannot foresee what products users are going to buy, how much they are going to spend and when the transactions will take place. Among all of those unpredictable future information, there is one thing that is actually predictable and can be used as a public key for encryption. What transactions are going to be included in each block are uncontrollable, but the header of each new block must satisfy the proof-of-work constraints (1) in order for the new block to be added into the main chain. Instead of encrypting with traditional keys, we encrypt with the proof-of-work constraints and any solution to the constraints will be a valid decryption key. In fact, both witness encryption [16] and indistinguishability Obfuscation ($i\mathcal{O}$) [15] can allow us to encrypt with such a constraint and decrypt with its solution.

To build our time-release protocol, we can use $i\mathcal{O}$ to obfuscate a program consisting of the proof-of-work constraints and a secret key, and let the obfuscator cough out the key when the checking of the constraints succeeds. More generally, we can let the program directly check the actual time from different time servers all over the world instead of bitcoin block chain and cough out the key when the time is due.

Witness encryption is in principle easier to achieve than program obfuscation in terms of security and efficiency. This is because the latter must hide the details of the program while the former does not have to. We will further discuss witness encryption in the next section. In order to encrypt with witness encryption, we write C program for the bitcoin proof-of-work constraints and translate it into CNF clauses by using the tool CBMC [10]. Then we can use the witness encryption to encode the CNF clauses to produce a ciphertext. Our C program (about 300 loc) implements SHA256 and the proof-of-work constraints (as described in Figure 3) for 5 linked block headers. We specify a recent block as a starting point for the timing, for example, the Block 350108 which was generated in March 2015 [1]. Since a new block is created every 10 minutes on average, we can customise the time-delay we want in the encryption by choosing the right number of blocks in the proof-of-work constraints, for example 5 blocks for 50 minutes delay from the specified starting point. Specifically, in the C program, in the "hashPrevBlock" field of the first block header, we put the hash value of the header of the Block 350108. The value of "hashPrevBlock" field of the second block header is the hash of the first block header, and so on. The other fields are unpredictable, so they are initialised with a nondeter-

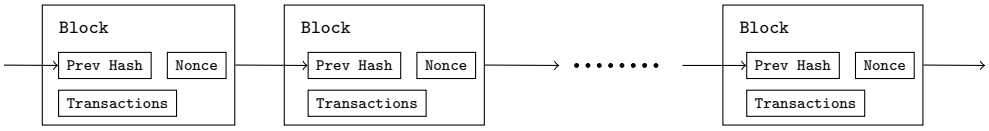

Figure 2: Bitcoin block chain

$$
\begin{aligned}
\underline{hash_1} &= \mathsf{SHA256}\left(\mathsf{SHA256}\left(ver_1, \underline{hash_0}, merkle_1, time_1, bit_1, nonce_1\right)\right) \leq target \\
\underline{hash_2} &= \mathsf{SHA256}\left(\mathsf{SHA256}\left(ver_2, \underline{hash_1}, merkle_2, time_2, bit_2, nonce_2\right)\right) \leq target \\
\underline{hash_3} &= \mathsf{SHA256}\left(\mathsf{SHA256}\left(ver_3, \underline{hash_2}, merkle_3, time_3, bit_3, nonce_3\right)\right) \leq target \\
&\cdots\cdots\cdots\cdots \\
\underline{hash_k} &= \mathsf{SHA256}\left(\mathsf{SHA256}\left(ver_k, \underline{hash_{k-1}}, merkle_k, time_k, bit_k, nonce_k\right)\right) \leq target
\end{aligned}
$$

Figure 3: Proof-of-work constraints

ministic value "nondet_uint()" which are handled as input variables in CBMC. Those nondeterministic values are the decryption key that will be generated by bitcoin network in future. More details about the source code can be found in Appendix A. Similar to all the previous computational complexity based time-release crypto, the accuracy of the time-delay is hard to achieve. The users do not have to wait for the bitcoin network to compute the values that satisfy the hash constraints; they can always compute by themselves if they have enough computing power. However, the bitcoin network possesses a huge computing power (currently around 30,000 Tera hashes per second) and it is unlikely for any individual to acquire such power alone.

CBMC [19, 10] is a Bounded Model Checker for C and C++ programs. CBMC transforms the program into static single assignment form [19], by unwinding loops (duplicating the loop body n times), expanding the (recursive) function calls and replacing the variables by bit vectors. This procedure produces two bit-vector equations: C (for the program constraints) and P (for the property). Then CBMC converts $C \wedge \neg P$ into CNF by adding intermediate variables. By construction, the CNF formula encodes the semantics of the program exactly: the CNF formula is satisfiable if and only if there is a program execution in the unwound program leading to an assertion violation.

By using CBMC, we transform our C program into the CNF formulas. The size of the resulting CNF formula is given in Figure 4. We use the tool Coprocessor [21], a CNF simplifier, for a quick simplification. We can see that the number of variables and clauses of the CNF formula increases linearly with the number of blocks. For the mining difficulty, the number of leading zeroes of the hash of block header is checked in an assertion statement. From the comparison of Figure 4a and Figure 4b, we can see that the change of mining difficulty does not affect the size of CNF formula. These figures will be useful for the plausibility discussion of the design of our protocol in Section 5.

4. WITNESS ENCRYPTION FOR SAT

We propose a new witness encryption for encrypting with CNF formula which generates smaller ciphertexts compared to the existing ones.

DEFINITION 4.1. *A* witness encryption *scheme [16] for an NP language L with corresponding witness relation R consists of the following two polynomial-time algorithms:*

- **Encryption.** *The algorithm* $\mathbf{Encrypt}\left(1^\lambda, x, M\right)$ *takes as input a security parameter* 1^λ*, an unbounded-length string x, and a message $M \in \{0,1\}$, and outputs a ciphertext CT.*
- **Decryption.** *The algorithm* $\mathbf{Decrypt}\left(CT, \mathbf{w}\right)$ *takes as input a ciphertext CT and a bit-vector $\mathbf{w}$, and outputs a message M or the symbol $\perp$.*

These algorithms satisfy the following two conditions:

- **Correctness.** *For any security parameter λ, for any $M \in \{0,1\}$ and for any $x \in L$ such that $R(x,w) = 1$ holds, we have that*

$$Pr\left[\mathbf{Decrypt}\left(\mathbf{Encrypt}\left(1^\lambda, x, M\right), w\right) = M\right] = 1$$

- **Soundness Security.** *For any PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{neg}(\lambda)$ *such that for any $x \notin L$, we have:*

$$\left|Pr\left[\mathcal{A}\left(\mathbf{Encrypt}\left(1^\lambda, x, 0\right)\right) = 1\right] - Pr\left[\mathcal{A}\left(\mathbf{Encrypt}\left(1^\lambda, x, 1\right)\right) = 1\right]\right| < \mathsf{neg}(\lambda)$$

The correctness states that an algorithm can decrypt if the instance x is in the language L (i.e., $x \in L$), and it knows a witness w such that $R(x,w) = 1$. The security states that if $x \notin L$, then no polynomial-time algorithm can decrypt.

4.1 SAT and Subset-Sum

We first briefly review the definition of SAT and SUBSET-SUM. Then we describe an intuitive textbook reduction from CNF-SAT to SUBSET-SUM.

We use notations $\mathbf{u}, \mathbf{v}, \mathbf{w}$ to represent integer vectors. We call a vector of n elements the n-vector. We write $\mathbf{0}_i$ for a vector constructed by i zeroes, and $\mathbf{1}_j$ for a vector constructed by j ones. We denote by $(\mathbf{u}, \mathbf{v})$ a vector obtained

CNF generated by CBMC			Simplified by Coprocessor		
$\#blocks$	$\#vars$	$\#clauses$	$\#vars$	$\#clauses$	$time(s)$
1	205,679	1,015,943	135,628	915,243	3.17
2	412,663	2,041,922	271,092	1,813,510	7.07
3	619,647	3,067,901	408,547	2,733,842	10.58
4	826,631	4,093,880	545,973	3,654,405	14.96
5	1,033,615	5,119,859	683,452	4,574,855	18.19

(a) Bitcoin mining difficulty: 64-bit leading zeroes

CNF generated by CBMC			Simplified by Coprocessor		
$\#blocks$	$\#vars$	$\#clauses$	$\#vars$	$\#clauses$	$time(s)$
1	205,689	1,016,025	135,429	916,279	3.16
2	412,683	2,042,086	272,764	1,824,906	7.01
3	619,677	3,068,147	410,180	2,745,325	10.56
4	826,671	4,094,208	547,661	3,671,326	12.60
5	1,033,665	5,120,269	685,198	4,595,175	15.84

(b) Bitcoin mining difficulty: 128-bit leading zeroes

Figure 4: Size of CNF clauses for bitcoin mining procedure

by appending vector $\mathbf{v}$ to vector $\mathbf{u}$. Let $\alpha := (\alpha_1, \cdots, \alpha_d)$ and $\mathbf{v} := (v_1, \cdots, v_d)$. We write $\alpha^{\mathbf{v}}$ for $\alpha_1^{v_1}\alpha_2^{v_2}\cdots\alpha_d^{v_d}$.

The *Boolean satisfiability problem* (SAT) is, given a formula, to check whether it is satisfiable.

Let B be a Boolean formula. A *literal* is either a variable x or the negation of a variable $\overline{x}$. A *clause* is a disjunction of literals, e.g., $C = x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4$. The formula B is said to be in *conjunctive normal form* (CNF) if it is a conjunction of clauses $C_1 \wedge C_2 \wedge \cdots \wedge C_m$. The SAT problem for CNF formulas is called CNF-SAT.

The Subset-Sum problem is: given a (multi)set Δ of integer vectors and a target integer vector $\mathbf{s}$, does there exist a subset of Δ such that the sum of its elements is equal to $\mathbf{s}$?

Reducing CNF-SAT to Subset-Sum. Assume a CNF formula of n variables $x_1, x_2 \ldots x_n$ and k clauses $C_1, C_2 \ldots C_k$. For each clause C_j, assume there are m_j literals in the clause. We consider the following textbook reduction from SAT to Subset-Sum:

1. For each variable x_i, construct two vectors $\mathbf{u}_{i,0}$ and $\mathbf{u}_{i,1}$ of $(n+k)$ integers as follows:
   - The i-th element of $\mathbf{u}_{i,0}$ and $\mathbf{u}_{i,1}$ is equal to 1
   - For $1 \le j \le k$, the $(n+j)$-th element of $\mathbf{u}_{i,0}$ is equal to 1 if $\overline{x_i}$ is in clause C_j
   - For $1 \le j \le k$, the $(n+j)$-th element of $\mathbf{u}_{i,1}$ is equal to 1 if x_i is in clause C_j
   - All other elements of $\mathbf{u}_{i,0}$ and $\mathbf{u}_{i,1}$ are 0
2. For each clause C_j, construct vectors $\mathbf{v}_{j,1}, \mathbf{v}_{j,2}, \cdots, \mathbf{v}_{j,m_j-1}$ of $n+k$ integers:
   - The $(n+j)$-th element of $\mathbf{v}_{j,1}, \mathbf{v}_{j,2}, \cdots, \mathbf{v}_{j,m_j-1}$ is equal to 1
   - All other elements of $\mathbf{v}_{j,1}, \mathbf{v}_{j,2}, \cdots, \mathbf{v}_{j,m_j-1}$ are 0
3. Finally, construct a sum vector $\mathbf{s}$ of $n+k$ integers:
   - For $1 \le j \le n$, the j-th element of $\mathbf{s}$ is equal to 1
   - For $1 \le j \le k$, the $(n+j)$-th element of $\mathbf{s}$ is equal to m_j.

Intuitively, the vector $\mathbf{u}_{i,0}$ corresponds to the negative occurrences of variable x_i in the formula while the vector $\mathbf{u}_{i,1}$ corresponds to its positive occurrences. The vectors $\mathbf{v}_{j,1}, \mathbf{v}_{j,2}, \cdots, \mathbf{v}_{j,m_j-1}$ for each clause C_j will sum to $m_j - 1$ at most, but to complete the sum m_j at least one will have to come from one of the $\mathbf{u}_{i,0}$ or $\mathbf{u}_{i,1}$ for $1 \le i \le n$.

Example 4.2. *$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3)$ is reduced into* Subset-Sum *as follows:*

	Variables				*Clauses*		
	x_1	x_2	x_3	x_4	C_1	C_2	C_3
$\mathbf{u}_{1,0}$	1					1	
$\mathbf{u}_{1,1}$	1				1		1
$\mathbf{u}_{2,0}$		1			1	1	
$\mathbf{u}_{2,1}$		1					1
$\mathbf{u}_{3,0}$			1				
$\mathbf{u}_{3,1}$			1			1	1
$\mathbf{u}_{4,0}$				1		1	
$\mathbf{u}_{4,1}$				1			
$\mathbf{v}_{1,1}$					1		
$\mathbf{v}_{2,1}$						1	
$\mathbf{v}_{2,2}$						1	
$\mathbf{v}_{2,3}$						1	
$\mathbf{v}_{3,1}$							1
$\mathbf{v}_{3,2}$							1
$\mathbf{s}$	1	1	1	1	2	4	3

Theorem 4.3. *The CNF formula is satisfiable iff subset sum exists.*

Proof. If there is a subsequence of integer vectors summing to the sum vector $\mathbf{s}$, this must use exactly one of each

of the pairs $(\mathbf{u}_{i,0}, \mathbf{u}_{i,1})$ (corresponding to each variable x_i being either false or true but not both) to make the first n columns correct. Also, each clause of C_j must have been satisfied by at least one variable, or the last k columns cannot be correct. Therefore, there is a satisfying assignment to $C_1 \wedge C_2 \wedge \cdots \wedge C_k$ if and only if there is a subsequence of the numbers that sums to $\mathbf{s}$. $\square$

4.2 Our construction

The (cryptographic) multilinear maps [7] only allow certain times of "multiplication", but not "division". We will make use of asymmetric multilinear maps in which groups are indexed by integer vectors. Suppose we have a $\mathbf{t}$-multilinear group family consisting of groups $\{G_{\mathbf{v}}\}_{\mathbf{v}}$ of the same order p and $\mathbf{v} \leq \mathbf{t}$, where $\mathbf{t}, \mathbf{v} \in \mathbb{Z}^{\ell}$ are positive integer vectors and the comparison between the vectors holds componentwise. The groups are equipped with a set of multilinear maps, $\mathsf{e}_{\mathbf{u},\mathbf{v}} : G_{\mathbf{u}} \times G_{\mathbf{v}} \to G_{\mathbf{u}+\mathbf{v}}$ for $\mathbf{u} + \mathbf{v} \leq \mathbf{t}$, satisfying $\mathsf{e}_{\mathbf{u},\mathbf{v}}\left(g_{\mathbf{u}}^{\alpha}, g_{\mathbf{v}}^{\beta}\right) = g_{\mathbf{u}+\mathbf{v}}^{\alpha\beta}$. We often omit the subscripts and just write e.

A direct encoding of Subset-Sum is given by Zhandry in [27]: for a collection m integers $v_1, ..., v_m$ and a target integer t, choose a random $\alpha \xleftarrow{R} \mathbb{Z}_p$, and encode each integer as $g_i^{\alpha^{v_i}}$ and the target as $g_T^{\alpha^t}$, with a map e such that $e\left(g_1, g_2, \cdots, g_m\right) = g_T$. This encoding is simple and effective, but it has to encode the same integers multiple times. We optimise the encoding for Subset-Sum by ensuring the same integer will only be encoded once, and our encoding is performed on the reduction of SAT to Subset-Sum, instead of directly on Subset-Sum.

Assume a CNF formula has n variables and k clauses, each clause C_j contains m_j literals and the total number of literals occurred is $m = \sum_{j=1}^{k} m_j$. The reduction from SAT to Subset-Sum mentioned in Section 4.1 is not linear in the size of $n+k$ by itself; it is of the size of the CNF formula, that is, the total number of literals occurred in all the clauses m. However, we can optimise the encoding for this reduction. The main ideas of our encoding and optimisation are:

1. Each non-zero element of the vector is encoded as a secret random, and an integer vector is encoded as the multiplications of these randoms.
2. Notice that for each clause C_j, the integer vectors $\mathbf{v}_{j,1}, \mathbf{v}_{j,2}, \cdots, \mathbf{v}_{j,m_j-1}$ are all the same. Instead of encoding them one-by-one, we can encode just one "seed" and the combination of these integers can be computed from the seed dynamically. However, we cannot allow the seed to be used for more than $m_j - 1$ times, otherwise the correctness of the reduction is not guaranteed. For this purpose, we associate each seed with a unique secret random as a counter to control usage of the seed.

Encrypt $\left(1^{\lambda}, x, M\right)$: Choose n randoms $a_1, a_2, \ldots, a_n$ for encoding variables, k randoms $b_1, \ldots, b_k$ for encoding clauses and another k randoms $c_1, \ldots, c_k$ for encoding the auxiliary counter. These randoms are kept secret.

1. Assume the vector $\mathbf{u}_{i,x_i} = (u_1, \cdots, u_n, u_{n+1}, \cdots, u_{n+k})$ with $x_i = 0, 1$. The encoding for $\mathbf{u}_{i,x_i}$ is U_{i,x_i}:
$$U_{i,x_i} := g_{\left(\mathbf{u}_{i,x_i}, \mathbf{0}_k\right)}^{\beta_{i,x_i}}$$
where
$$\beta_{i,x_i} := a_i \prod_{\substack{u_{n+j}=1 \\ 1 \leq j \leq k}} b_j$$
2. For $1 \leq j \leq k$, let $\mathbf{y}_j = (y_{j,1}, \cdots, y_{j,k})$ be a k-vector of zero except $y_{j,j} = 1$. We encode $\mathbf{v}_{j,1}$ as $(V_{j,0}, V_{j,1})$:
$$V_{j,0} := g_{\left(\mathbf{v}_{j,1}, \mathbf{y}_j\right)}^{b_j c_j}$$
$$V_{j,1} := g_{\left(\mathbf{0}_{n+k}, \mathbf{y}_j\right)}^{c_j}$$
Here $c_j, \mathbf{y}_j$ are used as a "counter" to guarantee that b_j in $V_{j,0}$ can be used for at most $m_j - 1$ times.
3. Correspondingly we extend the sum vector $\mathbf{s}$ with the auxiliary counter, then the sum becomes a $(n + 2k)$-vector
$$\mathbf{t} = (\mathbf{s}, m_1 - 1, \cdots, m_k - 1)$$
We encode the sum as
$$S := g_{\mathbf{t}}^{a_1 \cdots a_n b_1^{m_1} \cdots b_k^{m_k} c_1^{m_1 - 1} \cdots c_k^{m_k - 1}}$$
4. The ciphertext for encrypting a message M consists of $\mathsf{enc}\left(M, S\right)$ with
$$\{U_{i,0}, U_{i,1}\}_{1 \leq i \leq n} \cup \{V_{j,0}, V_{j,1}\}_{1 \leq j \leq k}$$

Decrypt $(CT, \mathbf{w})$: Given a witness $\mathbf{w} = (x_1, x_2, \cdots, x_n)$ which is an assignment for n variables. For each C_j, compute the number of literals that are evaluated to true and assume this number is z_j.

1. Compute the $m_j - z_j$ times mapping of $V_{j,0} = g_{\left(\mathbf{v}_{j,1}, \mathbf{y}_j\right)}^{b_j c_j}$
2. Compute the $z_j - 1$ times of mapping of $V_{j,1} = g_{\left(\mathbf{0}_{n+k}, \mathbf{y}_j\right)}^{c_j}$
3. Compute the mapping of the above two results to obtain W_j:
$$W_j := \mathsf{e}\left(g_{\left((m_j - z_j)\mathbf{v}_{j,1}, (m_j - z_j)\mathbf{y}_j\right)}^{b_j^{m_j - z_j} c_j^{m_j - z_j}}, g_{\left((z_j - 1)\mathbf{0}_{n+k}, (z_j - 1)\mathbf{y}_j\right)}^{c_j^{z_j - 1}}\right)$$
$$= g_{\left((m_j - z_j)\mathbf{v}_{j,1}, (m_j - 1)\mathbf{y}_j\right)}^{b_j^{m_j - z_j} c_j^{m_j - 1}}$$
4. Compute $\mathsf{e}\left(U_{1,x_1}, U_{2,x_2}, \cdots, U_{n,x_n}, W_1, \cdots, W_k\right)$ which will give the encoding for the sum.

Our encoding for the CNF formula of n variables and k clauses is of the size of $2n + 2k$ group elements and it is independent of the underlying implementation of multilinear maps. The multilinearity κ is $n + m - k$ consisting of n mapping operations among the variables and $m - k$ mapping operations among $\{V_{j,0}, V_{j,1}\}_{1 \leq j \leq k}$ since each pair of $(V_{j,0}, V_{j,1})$ will only be mapped for $m_j - 1$ times in total.

The current candidate multilinear maps [14, 11, 20] are not compact, that is, the size of elements in the groups depends on the multilinearity κ. We will further discuss this in the next section and we shall provide a method to balance the multilinearity level and the number of group elements in the ciphertext in order to compromise with the state-of-art implementation of multilinear maps.

EXAMPLE 4.4. *$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3)$ is encoded as below. Since all the vectors (except the sum) are sparse and only contain 0 and 1, for notational convenience, the group is indexed by the set of the positions of the non-zero elements, instead of the vector. For example, the set $\{1, 5, 7\}$ represents the vector $(1, 0, 0, 0, 1, 0, 1, 0, 0, 0)$.*

	Variables				*Clauses*			*Encoding*
	x_1	x_2	x_3	x_4	C_1	C_2	C_3	
	a_1	a_2	a_3	a_4	b_1	b_2	b_3	
$\mathbf{u}_{1,0}$	1					1		$g^{a_1 b_2}_{\{1,6\}}$
$\mathbf{u}_{1,1}$	1				1		1	$g^{a_1 b_1 b_3}_{\{1,5,7\}}$
$\mathbf{u}_{2,0}$		1			1	1		$g^{a_2 b_1 b_2}_{\{2,5,6\}}$
$\mathbf{u}_{2,1}$		1					1	$g^{a_2 b_3}_{\{2,7\}}$
$\mathbf{u}_{3,0}$			1					$g^{a_3}_{\{3\}}$
$\mathbf{u}_{3,1}$			1			1	1	$g^{a_3 b_2 b_3}_{\{3,6,7\}}$
$\mathbf{u}_{4,0}$				1		1		$g^{a_4 b_2}_{\{4,6\}}$
$\mathbf{u}_{4,1}$				1				$g^{a_4}_{\{4\}}$
$\mathbf{v}_{1,1}$					1			$g^{b_1 c_1}_{\{5,8\}}, g^{c_1}_{\{8\}}$
$\mathbf{v}_{2,1}$						1		$g^{b_2 c_2}_{\{6,9\}}, g^{c_2}_{\{9\}}$
$\mathbf{v}_{2,2}$						1		
$\mathbf{v}_{2,3}$						1		
$\mathbf{v}_{3,1}$							1	$g^{b_3 c_3}_{\{7,10\}}, g^{c_3}_{\{10\}}$
$\mathbf{v}_{3,2}$							1	
$\mathbf{s}$	1	1	1	1	2	4	3	$g^{a_1 a_2 a_3 a_4 b_1^2 b_2^4 b_3^3 c_1 c_2^3 c_3^2}_{(1,1,1,1,2,4,3,1,3,2)}$

Security analysis. Let $\alpha = (a_1, \cdots, a_n, b_1, \cdots, b_k, c_1, \cdots, c_k)$. Essentially, the above encoding for a vector $\mathbf{v}$ is of the form $g^{\alpha^{\mathbf{v}}}_{\mathbf{v}}$. However, this encoding does not directly work for the general SUBSET-SUM problem. For example, given $\{1\}$, there is no subset-sum for 3, but for the encoding we can obtain $g^{\alpha^3}_3$ by computing $\mathsf{e}\left(g^{\alpha^1}_1, g^{\alpha^1}_1, g^{\alpha^1}_1\right)$.

The above encoding works on a special class of SUBSET-SUM problem in which the given vectors occur sufficiently many times compared to the elements in the target sum. We adapt the *multilinear subset-sum Diffie-Hellman assumption* in [27] to *multilinear counting subset-sum Diffie-Hellman assumption*:

ASSUMPTION 4.5. *Our security assumption concerns a special class of* SUBSET-SUM *problem.*

Given a multi-set of vectors $\Delta = \{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ [1] with $\mathbf{v}_i \in \{0,1\}^d$, and a target sum vector $\mathbf{t} = (t_1, \cdots, t_d)$ of positive integers such that: for each $i \in I$, let $\mathbf{v}_i = (v_{i,1}, \cdots, v_{i,d})$, then $\ell_i \geq \min\{\, t_j \mid v_{i,j} = 1, 1 \leq j \leq d \,\}$.

Choose a vector of randoms $\alpha := (\alpha_1, \cdots, \alpha_d)$. If $\mathbf{t}$ cannot be represented as a subset-sum of elements from set Δ, then given $\left\{g^{\alpha^{\mathbf{v}_i}}_{\mathbf{v}_i}\right\}_{i \in I}$, $g^{\alpha^{\mathbf{t}}}_{\mathbf{t}}$ is indistinguishable from a random group element.

We name this assumption multilinear counting subset-sum Diffie-Hellman assumption.

THEOREM 4.6. *Our witness encryption for SAT is a sound scheme under multilinear counting subset-sum Diffie-Hellman assumption.*

PROOF. See Appendix B. □

We remark that our assumption depends on the challenge instance which means it is not instance independent. A framework for proving witness encryption schemes secure under instance independent assumptions is given in [17]. However, the proof techniques in [17] require large security parameters (e.g., for λ-bit security, the actual security parameter is $\lambda' = \lambda + \tilde{O}(n+k)$), and are subject to the recent attacks [9, 18] on current candidate multilinear maps. Our scheme achieves smaller ciphertext and is independent of the implementations of multilinear maps. We leave it as a future work to improve the security analysis to be based on instance independent assumptions.

5. INSTANTIATION AND PLAUSIBILITY

Since the breakthrough construction of Garg, Gentry and Halevi [14] in 2013, multilinear maps [14, 11, 20, 12] becomes a very active research area, as well as its cryptanalysis [9, 18, 26]. Current candidate multilinear maps [14, 11, 26, 12] are only approximate and do not satisfy the ideal model outlined in Section 4.2.

We instantiate our witness encryption with the CLT multilinear maps [11, 12] in order to estimate the efficiency and justify the plausibility of the design of our time-release protocol. We will show that the time-delay caused by efficiency of current candidate multilinear maps is essentially different and independent of the time-delay of bitcoin mining procedure. The attacks [9] on the original CLT maps [11] heavily rely on sufficiently many low-level encodings of zero which are usually not explicitly published in applications like witness encryption and indistinguishability obfuscation because whomever generates the encryption sets up the map and knows all the secret parameters. A tentative fix of the CLT maps is given in [12] which has the same CLT encodings but a different zero-testing procedure. The security analysis of the instantiation is not of the concern of this paper and we leave it as a future work. Our design of witness encryption is independent of the underlying implementations

[1] $(\mathbf{v}_i : \ell_i)$ means $\mathbf{v}_i$ occurs ℓ_i times in the multiset.

of multilinear maps. This means whenever there is a better implementation of multilinear maps (for better security assurances or better efficiency), we can simply swap it in.

The CLT maps generates N secret primes p_i and publishes $x_0 = \Pi_{i=1}^{N} p_i$, and also generates N small secret primes g_i. The message space is $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_N}$. For implementing the asymmetric multilinear maps, we select a series of random secret integers $z_j \bmod x_0$, $j = 1, \cdots, \ell$. For an index vector $\mathbf{v} = (v_1, \cdots, v_\ell)$, the encoding of a message $\mathbf{m} = (m_i) \in R$ relative to the index is then an integer c such that for all $1 \le i \le N$:

$$c \equiv \frac{r_i \cdot g_i + m_i}{z_1^{v_1} z_2^{v_2} \cdots z_\ell^{v_\ell}} \pmod{p_i}$$

for some small random integers r_i. Encodings can then be added and multiplied modulo x_0, as long as the noise r_i is such that $r_i \cdot g_i + m_i < p_i$ for each i. The encodings of group elements are noised and thus not unique. There is an extraction procedure that takes an encoding of a group element in the the target group and outputs a canonical representation.

According to the suggestions of parameters settings of the CLT maps [11, 12], we can take the size of group elements as $\Omega\left(\kappa^2 \lambda^3\right)$ for λ-bit security. The multilinearity κ equals to the number of multiplication operations.

In our scheme, the decryption involves $n + m - k$ mapping operations as analysed in Section 4.2. The computation of $g^{c_j^i}_{i(\mathbf{0}_{n+k}, \mathbf{y}_j)}$ from $V_{j,1} = g^{c_j}_{(\mathbf{0}_{n+k}, \mathbf{y}_j)}$ can be done in time $O(\log(i))$. Hence the decryption has time complexity $O\left(n + \sum_{j=1}^{k} \log(m_j)\right)$. However, this will not help reduce the multilinearity level when instantiating the scheme in the CLT maps and the multilinearity will still be $n + m - k$. To better suit the CLT maps, instead of encoding $\mathbf{v}_{j,1}$ as $\left(g^{b_j c_j}_{(\mathbf{v}_{j,1}, \mathbf{y}_j)}, g^{c_j}_{(\mathbf{0}_{n+k}, \mathbf{y}_j)}\right)$, we encode $\mathbf{v}_{j,1}$ as $2\lfloor \log(m_j) \rfloor$ elements:

$$g^{b_j c_j}_{(\mathbf{v}_{j,1}, \mathbf{y}_j)}, \quad g^{b_j^2 c_j^2}_{2(\mathbf{v}_{j,1}, \mathbf{y}_j)}, \quad g^{b_j^4 c_j^4}_{4(\mathbf{v}_{j,1}, \mathbf{y}_j)}, \quad \cdots, \quad g^{b_j^d c_j^d}_{d(\mathbf{v}_{j,1}, \mathbf{y}_j)}$$
$$g^{c_j}_{(\mathbf{0}_{n+k}, \mathbf{y}_j)}, \quad g^{c_j^2}_{2(\mathbf{0}_{n+k}, \mathbf{y}_j)}, \quad g^{c_j^4}_{4(\mathbf{0}_{n+k}, \mathbf{y}_j)}, \quad \cdots, \quad g^{c_j^d}_{d(\mathbf{0}_{n+k}, \mathbf{y}_j)}$$

where $d = 2^{\lfloor \log(m_j) \rfloor}$. As a result, this will keep the multilinearity as $n + \sum_{j=1}^{k} \log(m_j)$, instead of $n + \sum_{j=1}^{k} (m_j - 1)$.

The current candidate multilinear maps are far from practical. For the application of witness encryption, the current time for encryption and decryption is astronomical. However, this time-delay is essentially independent of the time-delay introduced in bitcoin mining procedure. The former is due to the state-of-art technology which will be improved in future, while the latter can be easily tuned. That is, the time for encryption and decryption will not be necessarily longer than the time of bitcoin mining. To illustrate this point, assume there is only one block involved in our time-releasing protocol. According to Figure 4 and the above analysis, the multilinearity κ is around 10^6. Let $\lambda = 256$. Then the size of group elements is around 10^{19}. The modulo multiplication $a \cdot b \bmod x_0$ has time complexity $O\left(\log^2(x_0)\right)$. Hence we estimate the time for multiplications in group of size 10^{19} from the time for multiplications in group of size 10^6 by the implementation in [11]. One multiplication in the group of size 10^{19} takes around 10^{24} seconds and the total decryption time is around $10^6 \cdot 10^{24} = 10^{30} (\approx 2^{100})$ seconds. Regardless of future scientific improvement of multilinear maps, just imaging our computing power is 10^{30} times faster at some point in future, then the decryption time will become 1 second, while the bitcoin mining difficulty (which is currently 64-bit leading zeroes) can be adjust to 164-leading zeroes in order to keep the speed of block generation at 10 minutes per block. The change of mining difficulty will not change the size of CNF formula as demonstrated in Figure 4, which will not affect the decryption time.

6. CONCLUSION

We have presented a new time-release protocol based on the bitcoin protocol. We build this protocol by witness encryption. The public key for encryption is the proof-of-work constraints and the decryption key are the unpredictable values in the future blocks. The unique feature of our protocol is that the decryption key will be automatically and publicly available in the bitcoin block chain when the time is due.

We have proposed a new witness encryption for CNF-SAT which achieves ciphertext size of $2n + 2k$ group elements, where n is the number of variables and k is the number of clauses of the CNF formula. Our witness encryption is based on an intuitive reduction of SAT to SUBSET-SUM problem. Our scheme is built on the framework of multilinear maps, but it is independent of the underlying implementations of multilinear maps.

7. REFERENCES

[1] Bitcoin Block Explorer. `https://blockexplorer.com`.

[2] Legal Threats Against Security Researchers. `http://attrition.org/errata/legal_threats/`.

[3] P. V. Ananth, D. Gupta, Y. Ishai, and A. Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *ACM CCS*, pages 646–658, 2014.

[4] M. Bellare and S. Goldwasser. Verifiable partial key escrow. In *Proc. 4th ACM Conference on Computer and Communications Security*, 1997.

[5] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. CRYPTO '01, pages 213–229. Springer-Verlag, 2001.

[6] D. Boneh and M. Naor. Timed commitments. In *CRYPTO*, pages 236–254, 2000.

[7] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002.

[8] J. Cathalo, B. Libert, and J. Quisquater. Efficient and non-interactive timed-release encryption. In *ICICS*, pages 291–303, 2005.

[9] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the Multilinear Map over the Integers. In *EUROCRYPT*, pages 3–12, 2015.

[10] E. Clarke, D. Kroening, and F. Lerda. A tool for checking ANSI-C programs. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, pages 168–176, 2004.

[11] J. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[12] J.-S. Coron, T. Lepoint, and M. Tibouchi. New multilinear maps over the integers. Cryptology ePrint Archive, Report 2015/162, 2015. `http://eprint.iacr.org/`.

[13] G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. EUROCRYPT'99, pages 74–89, 1999.

[14] S. Garg, C. Gentry, and S. Halevi. Candidate Multilinear Maps from Ideal Lattices. In *EUROCRYPT*, pages 1–17, 2013.

[15] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *FOCS*, pages 40–49, 2013.

[16] S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness Encryption and Its Applications. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 467–476, 2013.

[17] C. Gentry, A. B. Lewko, and B. Waters. Witness Encryption from Instance Independent Assumptions. In *Advances in Cryptology - CRYPTO 2014*, pages 426–443, 2014.

[18] Jean-Sébastien Coron and Tancrède Lepoint and Mehdi Tibouchi. Cryptanalysis of Two Candidate Fixes of Multilinear Maps over the Integers. `https://eprint.iacr.org/2014/975.pdf`.

[19] D. Kroening, E. Clarke, and K. Yorav. Behavioral consistency of C and Verilog programs using bounded model checking. In *Proceedings of DAC 2003*, pages 368–371, 2003.

[20] A. Langlois, D. Stehlé, and R. Steinfeld. GGHLite: More Efficient Multilinear Maps from Ideal Lattices. In *EUROCRYPT*, pages 239–256, 2014.

[21] Norbert Manthey. Riss 4.27 (with Coprocessor, Qprocessor, Mprocessor, ShiftBMC, Priss, Pcasso and BlackBox). `http://tools.computational-logic.org/content/riss427.php`.

[22] K. G. Paterson and E. A. Quaglia. Time-specific encryption. In *SCN*, pages 1–16, 2010.

[23] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, 1996. MIT/LCS/TR-684.

[24] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`.

[25] Timothy May. Time-release crypto. manuscript, February 1993.

[26] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH Map. `https://eprint.iacr.org/2015/301.pdf`.

[27] M. Zhandry. How to avoid obfuscation using witness prfs. *IACR Cryptology ePrint Archive*, 2014:301, 2014.

[28] J. Zimmerman. How to obfuscate programs directly. In *EUROCRYPT Part II*, pages 439–467, 2015.

APPENDIX
A. PSEUDO-CODE FOR PROOF-OF-WORK CONSTRAINTS

```
//verification of proof-of-work for the first block

 block1[20] = {
        nondet_uint(),            //version number
        0x5ffaaa03,               //hash of the block 350108
        0xf11aa3ee,
        0xd376d630,
        0x7ac05dc9,
        0x12d023bf,
        0x0ae3d1a3,
        0x00000000,
        0x00000000,
        nondet_uint(),            //merkle root of transactions
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),            //timestamp
        nondet_uint(),            //bits
        nondet_uint()             //nonce
 };

 hash1 = SHA256(SHA256(block1));            //hash of the first block,
                                            //we omit the implementation details for SHA256

 assert(!(hash1[7] == 0 && hash1[6] == 0)); //verifying the leading 64-bits are zeroes



//verification of proof-of-work for the second block

 block2[20] = {
        nondet_uint(),            //version number
        hash1[0],                 //hash of the first block
        hash1[1],
        hash1[2],
        hash1[3],
        hash1[4],
        hash1[5],
        hash1[6],
        hash1[7],
        nondet_uint(),            //merkle root of transactions
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),
        nondet_uint(),            //timestamp
        nondet_uint(),            //bits
        nondet_uint()             //nonce
 };

 hash2 = SHA256(SHA256(block2));              //hash of the second block

 assert(!(hash2[7] == 0 && hash2[6] == 0));   //verifying the leading 64-bits are zeroes
```

B. SECURITY ANALYSIS

Theorem 4.6. Our witness encryption for SAT is a sound scheme under multilinear counting subset-sum Diffie-Hellman assumption.

PROOF. Our witness encryption is for the SAT problem. The encoding described in Section 4.2 essentially extends the original reduction from SAT to SUBSET-SUM. Hence we will show this extension gives equivalent SUBSET-SUM problem and satisfy the condition in Assumption 4.5.

We describe this extended reduction as follows. Assume a CNF formula of n variables $x_1, x_2, \cdots, x_n$ and k clauses $C_1, C_2, \cdots, C_k$.

1. For each variable x_i, construct two vectors $\mathbf{u}'_{i,0}$ and $\mathbf{u}'_{i,1}$ of $(n+2k)$ integers as follows:
   - The i-th element of $\mathbf{u}'_{i,0}$ and $\mathbf{u}'_{i,1}$ is equal to 1
   - For $n+1 \leq j \leq n+k$, the j-th element of $\mathbf{u}'_{i,0}$ is equal to 1 if $\overline{x_i}$ is in clause C_{j-n}
   - For $n+1 \leq j \leq n+k$, the j-th element of $\mathbf{u}'_{i,1}$ is equal to 1 if x_i is in clause C_{j-n}
   - All other elements of $\mathbf{u}'_{i,0}$ and $\mathbf{u}'_{i,1}$ are 0
2. For each clause C_j, assume there are m_j literals in the clause C_j, construct vectors $\mathbf{v}'_{j,1}, \mathbf{v}'_{j,2}, \cdots, \mathbf{v}_{j,m_j-1}$ of $n+2k$ integers:
   - The $(n+j)$-th element and the $(n+k+j)$-th element of $\mathbf{v}'_{j,1}, \mathbf{v}'_{j,2}, \cdots, \mathbf{v}'_{j,m_j-1}$ are equal to 1
   - All other elements of $\mathbf{v}'_{j,1}, \mathbf{v}'_{j,2}, \cdots, \mathbf{v}'_{j,m_j-1}$ are 0
3. For each clause C_j, we add vectors $\mathbf{f}_{j,1}, \mathbf{f}_{j,2}, \cdots, \mathbf{f}_{j,m_j-1}$ of $n+2k$ integers as counters:
   - The $(n+k+j)$-th element of $\mathbf{f}_{j,1}, \mathbf{f}_{j,2}, \cdots, \mathbf{f}_{j,m_j-1}$ is equal to 1
   - All other elements of $\mathbf{f}_j$ is 0
4. Finally, construct a sum vector $\mathbf{t}$ of $n+2k$ integers:
   - For $1 \leq j \leq n$, the j-th element of $\mathbf{t}$ is equal to 1
   - For $1 \leq j \leq k$, the $(n+j)$-th element of $\mathbf{t}$ is equal to m_j.
   - For $1 \leq j \leq k$, the $(n+k+j)$-th element of $\mathbf{t}$ is equal to $m_j - 1$.

Compared to the original reduction in Section 4.1, this extended reduction appends k extra integers to the vectors which corresponds to the counter information. We use the notations in the encoding of Section 4.2 and let $\alpha = (a_1, \cdots, a_n, b_1, \cdots, b_k, c_1, \cdots, c_k)$. We encode a vector $\mathbf{v}$ as $g_{\mathbf{v}}^{\alpha^{\mathbf{v}}}$ and we use this encoding method to encode the above vectors. Comparing with the encoding in Section 4.2, we can easily see that

$$U_{i,0} = g_{\mathbf{u}'_{i,0}}^{\alpha^{\mathbf{u}'_{i,0}}} \text{ and } U_{i,1} = g_{\mathbf{u}'_{i,1}}^{\alpha^{\mathbf{u}'_{i,1}}} \text{ for } 1 \leq i \leq n$$
$$V_{j,0} = g_{\mathbf{v}'_{j,1}}^{\alpha^{\mathbf{v}'_{j,1}}} \text{ and } V_{j,1} = g_{\mathbf{f}_{j,1}}^{\alpha^{\mathbf{f}_{j,1}}} \text{ for } 1 \leq j \leq k$$

That is to say, encoding the above vectors generates the same encoding as the one described in Section 4.2.

Next we shall check the above SUBSET-SUM problem satisfy the condition in Assumption 4.5.

1. For $1 \leq i \leq n$, $\mathbf{u}'_{i,0}$ only occurs once.[2] Let $\mathbf{u}'_{i,0} = (u_1, \cdots, u_{n+2k})$. We can easily see that $\mathsf{min}\{\, t_j \mid u_j = 1, 1 \leq j \leq n+2k \,\} = 1$. Hence the condition holds.
2. The analysis for $\mathbf{u}'_{i,1}$ is similar.
3. The vectors $\mathbf{v}'_{j,1}, \mathbf{v}'_{j,2}, \cdots, \mathbf{v}'_{j,m_j-1}$ are the same vectors and the number of occurrence of this vector is $m_j - 1$. Only the $n+j$-th and $n+k+j$-th elements of those vectors are 1. The $n+j$-th element of the sum vector $\mathbf{t}$ is m_j and the $n+k+j$-th element of the sum vector $\mathbf{t}$ is $m_j - 1$. Hence the condition holds.
4. The vectors $\mathbf{f}_{j,1}, \mathbf{f}_{j,2}, \cdots, \mathbf{f}_{j,m_j-1}$ are the same vectors and the number of occurrence of this vector is $m_j - 1$. Only the $n+j+k$-th element of those vectors is 1. The $n+k+j$-th element of the sum vector $\mathbf{t}$ is $m_j - 1$. Hence the condition holds.

The soundness security of our witness encryption scheme follows immediately. □

[2]Note that the two vectors $\mathbf{u}'_{i,0}$ and $\mathbf{u}'_{i,1}$ might be the same. However, that means x_i and $\overline{x_i}$ occur at the same time in the same clauses. In this case, these clauses are definitely satisfiable and can be removed.

C. 3SAT TO EXACT-COVER

We give the textbook reduction from 3SAT to EXACT-COVER (the best reduction we can find) for the convenience of reader.

DEFINITION C.1 (EXACT-COVER).

- **Instance:** *a set X and a family $\mathcal{A}$ of subsets of X*
- **Decide:** *Is there an exact cover of X by $\mathcal{A}$?*

Reducing 3SAT to Exact-Cover. Let f be an instance of 3SAT, with variables $x_1, \cdots, x_n$ and clauses $f_1, \cdots, f_m$. We first construct a graph G from f by setting:

$$V(G) = \{ x_i \mid 1 \le i \le n \} \cup \{ \overline{x_i} \mid 1 \le i \le n \} \cup \{ f_j \mid 1 \le j \le m \}$$
$$E(G) = \{ x_i\overline{x_i} \mid 1 \le i \le n \} \cup \{ x_i f_j \mid x_i \in f_j \} \cup \{ \overline{x_i} f_j \mid \overline{x_i} \in f_j \}$$

where the notation $x_i \in f_j$ (resp. $\overline{x}_i \in f_j$) signifies that x_i (resp. $\overline{x}_i$) is a literal of the clause f_j. We then obtain an instance $(X, \mathcal{A})$ of the EXACT-COVER problem from this graph G by setting:

$$X = \{ f_j \mid 1 \le j \le m \} \cup E(G)$$
$$\mathcal{A} = \{ E(x_i) \mid 1 \le i \le n \} \cup \{ E(\overline{x_i}) \mid 1 \le i \le n \} \cup \{ \{f_j\} \cup F_j \mid F_j \subset E(f_j), 1 \le j \le m \}$$

where $E(x)$ denotes the set of edges incident to vertex x in the graph G.

It can be verified that the formula f is satisfiable if and only if the set X has an exact cover by the familiy of $\mathcal{A}$.

Remark. Although the above reduction does not explicitly refer to 3SAT, for a clause of length ℓ, the reduction would generate 2^ℓ sets. Hence, the CNF formula has to be reduced into 3CNF first to keep it as an polynomial reduction. Clearly, the number of sets in $\mathcal{A}$ is $2n + 7m$.