Extractable Witness Encryption and Timed-Release Encryption from Bitcoin

Jia Liu, Saqib A. Kakvi, Bogdan Warinschi

Department of Computer Science, University of Bristol

Abstract. We propose a new witness encryption based on Subset-Sum which achieves extractable security without relying on obfuscation and is more efficient than the existing ones. Our witness encryption employs multilinear maps of arbitrary order and it is independent of the implementations of multilinear maps. As an application, we construct a new timed-release encryption based on the Bitcoin protocol and extractable witness encryption. The novelty of our scheme is that the decryption key will be automatically revealed in the bitcoin block-chain when the block-chain reaches a certain length.

Keywords: timed-release encryption, Bitcoin, proof-of-work, witness encryption, SAT, Subset-Sum, multilinear maps, extractability

1 Introduction

Timed-release crypto aims to "send information into the future". This primitive, which was first suggested by May [37] in 1993 and then further studied by Rivest, Shamir and Wagner [35], enjoys many applications (electronic auctions, scheduled payment methods, sealed-bid auctions, lotteries). For example, this primitive can be used for responsible disclosure of vulnerabilities. Assume Alice wants to publish a paper about security flaws of some product, but the vendor of the product tries to prevent this publication. To mitigate this issue, Alice can publish an encrypted version of her paper by using a timed-release encryption scheme. In this way, her paper is only made available to the public after a certain period which gives the vendor a grace period to fix their products. After the grace period passed, the decryption key will be made available to the public and everyone can decrypt. During this waiting period, Alice cannot change her paper, and the vendor cannot prevent recovery of the decryption key and therefore the release of her findings.

Two distinct approaches have been used to solve the problem, one based on computational complexity [35, 6, 10] and another that uses a trusted third party (TTP) [21, 35, 9, 15, 34]. Informally, the latter approach relies on a strong trust assumption, the existence of a TTP who releases secrets at a specified time in the future. The former approach does not depend on any TTP but requires (whoever wants to decrypt) to spend a certain amount of computational effort to obtain the decryption key.

The main conceptual contribution of this paper is a novel timed-release encryption which eliminates the shortcomings of the aproaches above. Specifically, we do not require a trusted third party yet the decryption key will be (unavoidably) realeased at a later time. The trick is that we rely on an existing system, namely Bitcoin and employ the computational effort spent by miners towards releasing decryption keys. Roughly speaking, the encryption key is the state of the Bitcoin ledger at the moment of encryption; the decryption key is any valid extension of this ledger with transaction blocks. Notice that on one hand the encrypted information is never lost as anyone with sufficient computational power can recover a decryption key. On the other hand, such a key will eventually be released by the Bitcoin miner community.

Our timed-release encryption scheme is built on top of the Bitcoin mechanism and a generic witness encryption scheme that is extractable. To the best of our knowledge, existing WE schemes [25, 27, 39] do not have efficient extraction methods (extractors for these schemes appear to be super-polynomial). An exception is the scheme of Bellare and Hoang [7] which defers the issue of extractability to its underlying obfuscation scheme. Our second main contribution is a new witness encryption scheme, which achieves extractable security without using obfuscation and which is more efficient than the previously known schemes.

1.1 Related Work

Witness encryption Witness encryption, originally proposed in [25], allows one to encrypt to an instance, x, of an NP language and decrypt using any witness w that x is in the language. The construction in [25] for witness encryption is based on the Exact-Cover problem and multilinear maps [11]. The security of the construction in [25] is called soundness security and is based on the "Decision Graded Encoding No-Exact-Cover Problem". Informally soundness security says that if the instance has no exact-cover, the ciphertext reveals no information about the plaintext. A proof framework is proposed in [26] for proving soundness security under instance independent assumptions.

Soundness security has several interesting applications [25, 7], but it is not sufficient for the general applications and in particular it is not sufficient for our application of timed-release encryption. A stronger notion called extractable security proposed in [27] informally says that when the adversary can distinguish two ciphertext encrypted with the same problem instance but different messages, then it must know a witness of the instance. A uniform variant of extractable security without auxiliary inputs is given in [7], but it is also not sufficient for our application of timed-release encryption. An extractable witness PRF is constructed in [39] by assuming the underlying Subset-Sum encoding is extractable.

Concurrent work The idea of building time-lock encryption from Bitcoin and witness encryption was described in Jager's concurrent work [28] and was also independently mentioned by Miller in a public chat room [32]. In our work we use some of the formal definitions proposed by Jager [28]. In particular the syntax and security of timed-release encryption are essentially those of [28]. Our model for a secure timer (a formalization of the guarantees for the Bitcoin backbone protocol) is a refinement of the model of a "computational reference clock" from [28].

There are two key differences between our work and that of Jager [28]:

1. The security of the time-lock encryption in [28] is based on the definition of extractable witness encryption given in [7]. However, the extractable witness encryption in [7] considers a uniform adversary without any auxiliary input. This definition appears not to suffice for the security of the time-lock encryption [28] because the adversary of the time-lock encryption has access to an external oracle. This has been pointed out to Jager through a private communication in September 2015. In comparison, in our work, we give a definition of extractable security in an oracle model and we show this definition is sufficient for our application of timed-release encryption.
2. The extractable witness encryption [27, 7] has no known instantiations except it is shown in [7] that the extractable obfuscation implies the extractable witness encryption. We construct a new witness encryption scheme and prove its extractable security in the generic model of the multilinear maps.

Other work on Bitcoin Several papers [3, 4, 40, 8, 30] use Bitcoin deposits as a penalty to facilitate fairness in multiparty computation. Security properties of Bitcoin backbone protocol, such as common prefix and chain quality, are formally analysed in [22]. The use of Bitcoin as a source of publicly-verifiable randomness is formalised in [13].

1.2 Our Contribution

Extractable witness encryption We first propose a new extractable witness encryption scheme based on Conjunctive Normal Form Satisfiability (CNF-SAT). We prove the extractable security of this in the Idealised Graded Encoding Model [14, 5, 41], with an intermediate reduction to a *special* SUBSET-SUM problem. To the best of our knowledge, this is the first construction of witness encryption to achieve extractable security, without the use of obfuscation.

This special SUBSET-SUM problem is: given a multi-set of positive integer vectors $\varDelta = \{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ where $(\mathbf{v}_i : \ell_i)$ means $\mathbf{v}_i$ occurs ℓ_i times in the multiset and a target sum-vector $\mathbf{s}$ such that $(\ell_i + 1)\mathbf{v}_i \nleq \mathbf{s}$ and $\mathbf{v}_i$ are pairwise-distinct, to decide whether there exists a subset of $\varDelta$ that can sum up to $\mathbf{s}$. The side condition $(\ell_i + 1)\mathbf{v}_i \nleq \mathbf{s}$ is to guarantee the encoding of each integer vector $\mathbf{v}_i$ can only be used for at most ℓ_i times, in order to keep consistency between the encoding of the SUBSET-SUM and the original SUBSET-SUM problem. Our encoding is based on an asymmetric variant of multilinear maps where the groups are indexed by the integer vectors [23, 5, 12, 41]. Suppose a $\mathbf{u}$-linear map on groups $\{\mathbb{G}_{\mathbf{w}}\}_{\mathbf{w}}$ with $\mathbf{w} \leq \mathbf{u}$ (component-wise comparison). The pairing operation $\mathsf{e}_{\mathbf{w},\mathbf{w}'}$ maps $\mathbb{G}_{\mathbf{w}} \times \mathbb{G}_{\mathbf{w}'}$ into $\mathbb{G}_{\mathbf{w}+\mathbf{w}'}$ with $\mathbf{w} + \mathbf{w}' \leq \mathbf{u}$ by computing $\mathsf{e}_{\mathbf{w},\mathbf{w}'}\left(g_{\mathbf{w}}^{a}, g_{\mathbf{w}'}^{b}\right) = g_{\mathbf{w}+\mathbf{w}'}^{ab}$. The vectors in $\varDelta = \{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ is encoded as $\left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i \in I}$ and the target vector is encoded as $g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}$. Suppose the subset-sum exists, that is $\sum_{i\in I} b_i\mathbf{v}_i = \mathbf{s}$ with b_i positive integers and $b_i \leq \ell_i$. Then we compute the encoding of the target sum as below

$$g_{\mathbf{s}}^{\alpha^{\mathbf{s}}} = \mathsf{e}(\underbrace{g_{\mathbf{v}_1}^{\alpha^{\mathbf{v}_1}}, \cdots, g_{\mathbf{v}_1}^{\alpha^{\mathbf{v}_1}}}_{b_1}, \underbrace{g_{\mathbf{v}_2}^{\alpha^{\mathbf{v}_2}}, \cdots, g_{\mathbf{v}_2}^{\alpha^{\mathbf{v}_2}}}_{b_2}, \cdots, \underbrace{g_{\mathbf{v}_{|I|}}^{\alpha^{\mathbf{v}_{|I|}}}, \cdots, g_{\mathbf{v}_{|I|}}^{\alpha^{\mathbf{v}_{|I|}}}}_{b_{|I|}})$$

In this way, each vector $\mathbf{v}_i$ only needs to be encoded once and the multiplication of the same encoding is in logarithm time which gains efficiency. To encrypt with any NP language, we present a reduction from CNF-SAT to our special Subset-Sum problem.

We prove our encoding achieves extractability in the generic model of multilinear maps. The main technique is to construct an efficient extractor to extract a witness from the adversary's group operations. Constructing a witness extractor for the existing witness encryption schemes [25, 27, 39] appears to be super-polynomial because of the expansion of adversary's query polynomial. Interestingly extractable witness encryption with arbitrary auxiliary inputs might be unattainable [24]. The counter example is constructed as follows: suppose the sampler has $(x, w) \in R$, then the sampler obfuscates the decryption algorithm of witness encryption $z = \mathcal{O}(\mathsf{WE.Dec}(w, \cdot))$ and gives z to the adversary; the adversary can decrypt $\mathsf{WE.Enc}(x, m)$ by using the backdoor z and does not have to know w. However, for our application of timed-release encryption, we do not need the extractable security with arbitrary auxiliary inputs. Instead, we define extractable security in an oracle model. The oracle is used to model the Bitcoin block-chain. The extractability is possible to achieve for most of the non-artificial oracles. In particular, when the oracle is instantiated with a decentralised cryptocurrency such as Bitcoin, this kind of backdoor is unlikely to exist since it is believed that no one can have a witness w in advance for each instance x. After all, the Bitcoin block-chain has been publicly available since 2009 and any adversary of any crypto scheme in the real world actually has access to it.

Efficiency comparison Assume a CNF formula has n variables and k clauses, and each clause C_j contains m_j literals. Let $m = \sum_{j=1}^{k} m_j$ be the total number of literals occurred. The ciphertext size of our witness encryption scheme is $2n + 2k + 1$ group elements. The evaluation time is $n + \mathcal{O}\big(k \log \frac{m}{2k}\big)$. The multilinearity level is $n + m - k$ and can be optimised to $n + \mathcal{O}\big(k \log \frac{m}{2k}\big)$.

The efficiency of encoding CNF-SAT in [25] depends on the reduction which is unspecified in [25]. As far as we know, the best reductions from CNF-SAT to Exact-Cover is CNF-SAT → 3-CNF-SAT → Exact-Cover (The details of the second reduction can be found in Appendix E). However, 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 $n' = \mathcal{O}(m)$ and $k' = \mathcal{O}(m)$ while m is $n \cdot k$ in the worst case. The reduction from 3-CNF-SAT to Exact-Cover generates an instance of size $2n' + 7k' + 1$. Hence the encoding produces $\mathcal{O}(m)$ group elements and the evaluation time and multilinearity level are also $\mathcal{O}(m)$ with $m = n \cdot k$ in the worst case.

There are three instantiations of witness encryption for CNF formulas in [26]. Two of them are specific to the composite order multilinear groups. The prime-order groups are usually more natural and result in simpler security assumptions. The conversion from composite-order construction to prime-order multilinear groups (or more generally, groups of arbitrary order) is very expensive and results in a ciphertext of $O(n^5k^2)$ group elements.

A direct encoding for SUBSET-SUM is given in [39] which encodes every integer vector in a different group. In comparison, our encoding for SUBSET-SUM only encodes the same integer vector once. As a result, the encoding of the CNF formula is of the size of $n+m-k$ group elements which is $O(n \cdot k)$ in the worst case. The multilinearity level and the evaluation time are both $n+m-k$.

Timed-release encryption from Bitcoin Having built our witness encryption scheme, we propose a new timed-release encryption based on the Bitcoin protocol and witness encryption. The novelty of our protocol is that the Bitcoin network does the required amount of computational work to recover the decryption key and the decryptor in our protocol does not have to invest the computational effort. In the traditional timed-release crypto, the actual decryption time depended 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 publicly available in the Bitcoin block-chain. Moreover, we do not modify the existing Bitcoin protocol. The integration with the Bitcoin protocol is seamless and the Bitcoin protocol functions like a public time clock.

The basic idea of our timed-release encryption is to extract a "public key" from Bitcoin block-chain for encryption today; the decryption key consists of the unpredictable information in the Bitcoin blocks (e.g., transactions, nonces, hashes) that will only be determined in future. 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:

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

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 the Bitcoin network as the next head of block-chain, its block header must satisfy the above constraint.

Our timed-release encryption encrypts to a recent block t and a length ℓ of proof-of-work constraints. The ℓ subsequent valid blocks of t will provide a solution to the constraints. A new block in the Bitcoin network is created every 10 minutes on average. Thus encrypting with ℓ blocks will give roughly $10 \cdot \ell$ minutes delay before the release of the information. After those blocks are generated, everyone can decrypt by using the information from those blocks. Hence the "public key" for encryption is the proof-of-work constraints. Technically we instantiate our timed-release encryption by our witness encryption which allows us to encrypt with such a constraint and decrypt with its solution.

1.3 Outline

The rest of this paper proceeds as follows. In Section 2, we propose our new witness encryption scheme. In Section 3, we provide a short overview of the Bitcoin protocol and analyse the success rate of Bitcoin mining. In Section 4, we propose our new timed-release encryption from the Bitcoin and witness encryption. The paper concludes in Section 5. The discussion about the instantiation of our timed-release encryption can be found in Appendix F.

2 Extractable witness encryption

2.1 Witness encryption

Witness encryption was originally proposed by Garg, Gentry, Sahai and Waters [25]. 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.

Definition 1 (Witness encryption (WE) [25]). *A* witness encryption *scheme for an NP language L with corresponding witness relation R_L consists of the following two polynomial-time algorithms:*

- $\mathsf{WE.Enc}\big(1^\lambda, x, m\big)$ *is an encryption algorithm that takes as input a security parameter 1^λ, an unbounded-length string x, and a message $m \in \mathbb{M}$, and outputs a ciphertext c.*
- $\mathsf{WE.Dec}(c, w)$ *is an decryption algorithm that takes as input a ciphertext c and a bit-vector w, and outputs a message m or the symbol $\perp$.*
- **Correctness.** *For any $x \in L$ such that $(x, w) \in R_L$, we have that*

$$Pr\big[\mathsf{WE.Dec}\big(\mathsf{WE.Enc}\big(1^\lambda, x, m\big), w\big) = m\big] = 1$$

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 $(x, w) \in R_L$.

A strong security notion for witness encryption is called extractable security which is originally proposed in [27]. The extractability says that when the adversary can distinguish two ciphertext encrypting different messages by using the same instance, then it must know a witness of the instance. Below we give a uniform variant of extractable security in an oracle model. We will show that this definition is sufficient for our application of timed-release encryption. In the next Section 2, we propose a novel witness encryption scheme and we prove our new scheme achieves the extractable security.

Definition 2 (Extractable security). *A witness encryption scheme for a language $L \in NP$ is $(t, t', q, q'; \varepsilon, \varepsilon')$-secure w.r.t. an oracle $O(x, \cdot)$ if for any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that run in time t and queries $O(x, \cdot)$ for at most q times,* *there exists an extractor E that runs in time t' and queries $O(x,\cdot)$ for at most q' times, such that for all $x \in \{0,1\}^*$, the following holds:*

$$\textit{if} \quad Pr\left[\begin{array}{l} (m_0, m_1, St) \leftarrow \mathcal{A}_0^{O(x,\cdot)}\big(1^\lambda, x\big) \\ b \leftarrow \{0,1\}\,;c \leftarrow \mathsf{WE.Enc}\big(1^\lambda, x, m_b\big) \; : b = b' \\ b' \leftarrow \mathcal{A}_1^{O(x,\cdot)}(x, m_0, m_1, c, St) \end{array}\right] > \varepsilon$$

$$\textit{then} \quad Pr\Big[w \leftarrow E^{O(x,\cdot)}\big(1^\lambda, x\big) : (x,w) \in R_L\Big] > \varepsilon'$$

The above definition states that if an adversary that runs in time t and queries the oracle for at most q times can break the encryption scheme with probability more than ε, then there exists an extractor E that runs in time t' and extracts a witness with probability more than ε'.

In the above definition, we give the adversary access to an oracle $O(x,\cdot)$. We do not specify what the oracle does except it is related to the instance x. In the next section, we shall prove the extractable security w.r.t. an oracle in the generic model of multilinear maps. This is due to the fact that we have our proof in an idealised model, which allows us to circumvent the impossibility result [24].

2.2 Extractable witness encryption from Subset-Sum

In this section, we propose a construction for extractable witness encryption from a special Subset-Sum problem and we prove the extractable security in the generic model of multilinear maps. We will show that the CNF-SAT can be reduced to this special Subset-Sum problem in the next section.

We use notations $\mathbf{u}, \mathbf{v}, \mathbf{w}$ to represent integer vectors. We call a vector of n elements the n-vector. We define $\mathbf{u} \leq \mathbf{v}$ as the component-wise comparison. We denote by $\{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ a multi-set in which the element $\mathbf{v}_i$ occurs ℓ_i times and $\mathbf{v}_1, \cdots, \mathbf{v}_{|I|}$ are pairwise-distinct. Let $\alpha := (\alpha_1, \cdots, \alpha_d)$ and $\mathbf{v} := (v_1, \cdots, v_d)$. We write $\alpha^{\mathbf{v}}$ for $\alpha_1^{v_1} \alpha_2^{v_2} \cdots \alpha_d^{v_d}$.

Our witness encryption scheme makes use of asymmetric multilinear maps in which groups are indexed by integer vectors [23, 12, 41]. Suppose we have a $\mathbf{s}$-multilinear group family consisting of groups $\{G_{\mathbf{v}}\}_{\mathbf{v}}$ of the same order p and $\mathbf{v} \leq \mathbf{s}$, where $\mathbf{s}, \mathbf{v} \in \mathbb{Z}^\ell$ are positive integer vectors and the comparison between the vectors holds component-wise. 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 $\mathbf{u} + \mathbf{v} \leq \mathbf{s}$, satisfying $\mathsf{e}_{\mathbf{u},\mathbf{v}}\big(g_{\mathbf{u}}^{\alpha}, g_{\mathbf{v}}^{\beta}\big) = g_{\mathbf{u}+\mathbf{v}}^{\alpha\beta}$. We often omit the subscripts and just write e.

The original Subset-Sum problem is: given a (multi)set of integer vectors and a target integer vector $\mathbf{s}$, does there exist a subset of the integer vectors such that the sum of its elements is equal to $\mathbf{s}$? To achieve extractability in witness encryption, our encoding will be performed on a special Subset-Sum problem defined as below:

- **Instance**: given a multi-set of d-vectors $\Delta = \{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ of positive integers, and a sum d-vector $\mathbf{s}$ of positive integers such that $(\ell_i + 1)\mathbf{v}_i \not\leq \mathbf{s}$ for each $i \in I$.

– **Decide**: is $\sum_{i\in I} b_i\mathbf{v}_i = \mathbf{s}$ for some integers $0 \le b_i \le \ell_i$ with $i \in I$?

Construction 1 (Extractable witness encryption) *Suppose x is an instance of the above special* Subset-Sum *problem and we use the above notations in the following discussion. We construct a witness encryption scheme as below:*

– $\mathsf{WE.Enc}\big(1^\lambda, x, m\big)$*: run* $\mathsf{param} \leftarrow \mathcal{G}\big(1^\lambda, \Delta, \mathbf{s}\big)$ *to get the description of 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* $\mathbf{u}+\mathbf{v} \le \mathbf{s}$*, together with group generators* $\{g_{\mathbf{v}}\}_{\mathbf{v}\le\mathbf{s}}$*. Choose a vector of randoms* $\alpha := \langle \alpha_1, \cdots, \alpha_d\rangle$ *and a random r. The ciphertext is* $c := \Big(\mathsf{param}, \big\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\big\}_{i\in I}, m \cdot g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}\Big)$*.*

– $\mathsf{WE.Dec}(c, \mathbf{w})$*: let* $\mathbf{w} := (b_1, b_2, \cdots, b_{|I|})$*. Compute the decryption key by*

$$K := \mathsf{e}(\underbrace{g_{\mathbf{v}_1}^{\alpha^{\mathbf{v}_1}}, \cdots, g_{\mathbf{v}_1}^{\alpha^{\mathbf{v}_1}}}_{b_1}, \underbrace{g_{\mathbf{v}_2}^{\alpha^{\mathbf{v}_2}}, \cdots, g_{\mathbf{v}_2}^{\alpha^{\mathbf{v}_2}}}_{b_2}, \cdots, \underbrace{g_{\mathbf{v}_{|I|}}^{\alpha^{\mathbf{v}_{|I|}}}, \cdots, g_{\mathbf{v}_{|I|}}^{\alpha^{\mathbf{v}_{|I|}}}}_{b_{|I|}})$$

If $\sum_{i\in I} b_i\mathbf{v}_i = \mathbf{s}$*, then* $K = g_{\sum_{i\in I} b_i\mathbf{v}_i}^{\alpha^{\sum_{i\in I} b_i\mathbf{v}_i}} = g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}$*.*

In the above construction, each vector $\mathbf{v}_i$ is only encoded once as a group element $g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}$. The multiple usage of $\mathbf{v}_i$ corresponds to the multiple pairing of $g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}$. However, we cannot allow the encoded element to be used for more than ℓ_i times. This is why we have the side condition $(\ell_i + 1)\mathbf{v}_i \not\le \mathbf{s}$. If $g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}$ is paired for more than ℓ_i times, then the group index will be no longer smaller than the index of the target group $\mathbf{s}$. Note that the encoding described above does not directly work for the general Subset-Sum problem. For example, given $\{1\}$, there is no subset-sum for 3, but we can obtain the encoding $g_3^{\alpha^3}$ for the sum by computing $\mathsf{e}\Big(g_1^{\alpha^1}, g_1^{\alpha^1}, g_1^{\alpha^1}\Big)$. The problem is caused by the fact that the encoding $g_1^{\alpha^1}$ can be used for multiple times but the element 1 can only be used for at most once in the Subset-Sum instance.

Theorem 1. *Our Construction 1 of witness encryption achieves extractable security with* $t' = \mathsf{poly}(t \cdot \lambda)$ *and* $\varepsilon' = 2\varepsilon - 1$ *and* $q' \le q$ *in the generic model of multilinear maps.*

Proof (Sketch). Due to the space limitation, the full proof can be found in Appendix C. The multilinear map oracle gives the adversary the handles for $\big\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\big\}_{i\in I}$ in order to track the adversary's group operations. The oracle answers the queries for adding two handles on the groups of the same index level and for pairing two handles into a new group with an index level lower or equal to the top index level $\mathbf{s}$. Note that the Subset-Sum instance is encoded in the index of the group which enables us to efficiently extract a subset sum from the adversary's query polynomial of the handles without expanding the polynomial into monomials. (The expansion of a polynomial is in general super-polynomial which seems to be the reason that extracting a witness for the existing schemes [25, 27, 39] is difficult.) If the adversary can construct an element from group $G_{\mathbf{s}}$ by using elements in group $\{G_{\mathbf{v}_i}\}_{i\in I}$, then we can efficiently extract a witness; otherwise the elements $\big\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\big\}_{i\in I}$ and the element $g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}$ are independently distributed from the adversary's point of view.

Our special Subset-Sum problem can directly encode the Exact-Cover problem by representing each set of the instance of Exact-Cover as a vector and the side condition holds because each set can be used for at most once. This encoding converts the witness encryption scheme [25] into an extractable witness encryption scheme.

A weaker security guarantee for witness encryption is the soundness security. The soundness security states that if $x \notin L$ then no polynomial-time algorithm can decrypt. An alternative definition for soundness security called adaptive soundness is given in [7]. In fact, the extractable security implies the soundness security since the probability that the extractor can extract a witness is 0 when $x \notin L$. We also give a discussion about the soundness security of our witness encryption scheme in the Appendix D.

Since the breakthrough construction of Garg, Gentry and Halevi [23] in 2013, multilinear maps [23, 19, 31, 20] becomes a very active research area, as well as its cryptanalysis [16, 29, 38, 17]. Our design of witness encryption is independent of the underlying implementations of multilinear maps. In particular, our scheme is not susceptible to the zeroising attacks, as we do not publish low level zero encodings.

2.3 Reducing CNF-SAT to Subset-Sum

In this section, we show how to construct an extractable witness encryption for any NP language. This is achieved by constructing an intuitive reduction from an instance of CNF-SAT to an instance of our special Subset-Sum. This results in a more efficient encoding for CNF formulas compared to the encoding in the existing witness encryption schemes.

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.

Definition 3 (Reduction from CNF-SAT to special Subset-Sum). *Assume a CNF formula has n variables $x_1, x_2, \cdots, x_n$, and k clauses $C_1, C_2, \cdots, C_k$, each clause C_j contains m_j literals. The reduction to an instance of special* Subset-Sum *is performed as below:*

1. *For each variable x_i with $1 \leq i \leq n$, 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 1*
   - *For $1 \leq j \leq k$, the $(n+j)$-th element of $\mathbf{u}_{i,0}$ is 1 if $\overline{x_i}$ is in clause C_j*
   - *For $1 \leq j \leq k$, the $(n+j)$-th element of $\mathbf{u}_{i,1}$ is 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 with $1 \leq j \leq k$, 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 construct vectors $\mathbf{z}_{j,1}, \mathbf{z}_{j,2}, \cdots, \mathbf{z}_{j,m_{j-1}}$ of $n+2k$ integers as counters:*
   - *The $(n+k+j)$-th element of $\mathbf{z}_{j,1}, \mathbf{z}_{j,2}, \cdots, \mathbf{z}_{j,m_{j-1}}$ is equal to* 1
   - *All other elements of $\mathbf{z}_j$ is* 0
4. *Finally, construct a target sum vector $\mathbf{s}$ of $n+2k$ integers:*
   - *For $1 \leq j \leq n$, the j-th element of $\mathbf{s}$ is equal to* 1
   - *For $1 \leq j \leq k$, the $(n+j)$-th element of $\mathbf{s}$ is equal to m_j.*
   - *For $1 \leq j \leq k$, the $(n+k+j)$-th element of $\mathbf{s}$ is equal to $m_j - 1$.*

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 $1 \leq i \leq n$, which means the clause has to be satisfied by some literals. An example for explaining this reduction is given in the following example.

Example 1. $(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3)$ is encoded in Figure 1. The assignment $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$ evaluates the formula to true, and it corresponds to the subset sum $\mathbf{u}_{1,1} + \mathbf{u}_{2,0} + \mathbf{u}_{3,1} + \mathbf{u}_{4,0} + \mathbf{v}_{2,1} + \mathbf{v}_{2,2} + \mathbf{v}_{3,1} + \mathbf{z}_{1,1} + \mathbf{z}_{2,1} + \mathbf{z}_{2,2} + \mathbf{z}_{3,1} = \mathbf{s}$.

We shall analyse that the above reduction transforms an instance of CNF-SAT into an instance of our special Subset-Sum, that is the side-condition $(\ell_i + 1)\mathbf{v}_i \not\leq \mathbf{s}$ is satisfied. Each vector $\mathbf{u}_{i,b}$ can occur for at most once in the sum, otherwise the i-th element of the sum will be bigger than 1. The vectors $\mathbf{v}_{j,1}, \cdots, \mathbf{v}_{j,m_j-1}$ are the same vectors and at most $m_j - 1$ of them can occur in the sum, otherwise the $n+k+j$-th element of the sum will be bigger than $m_j - 1$. Similarly the vectors $\mathbf{z}_{j,1}, \cdots, \mathbf{z}_{j,m_j-1}$ are the same and at most $m_j - 1$ of them can be used in the sum otherwise the $n+k+j$-th element of the sum will be bigger than $m_j - 1$.

Theorem 2. *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 elements of the sum correct. Also, each clause of C_j must have been satisfied by at least one variable, or the next k elements of the sum cannot be big enough. The last k elements of the vectors are the auxiliary counters that will be used for encoding. 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 $\mathbf{s}$.

The reduction of the CNF formula to an instance of the special Subset-Sum consists of $2n + m$ vectors. However, when we encrypt with Construction 1, the vectors $\mathbf{v}_{j,1}, \cdots, \mathbf{v}_{j,m_j-1}$ are encoded as one group element $g_{\mathbf{v}_{j,1}}^{\alpha^{\mathbf{v}_{j,1}}}$ since they are

	Variables				*Clauses*			*Counter*		
	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			1		
$\mathbf{v}_{2,1}$						1			1	
$\mathbf{v}_{2,2}$						1			1	
$\mathbf{v}_{2,3}$						1			1	
$\mathbf{v}_{3,1}$							1			1
$\mathbf{v}_{3,2}$							1			1
$\mathbf{z}_{1,1}$								1		
$\mathbf{z}_{2,1}$									1	
$\mathbf{z}_{2,2}$									1	
$\mathbf{z}_{2,3}$									1	
$\mathbf{z}_{3,1}$										1
$\mathbf{z}_{3,2}$										1
$\mathbf{s}$	1	1	1	1	2	4	3	1	3	2

Fig. 1: Reducing $(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (x_1 \vee x_2 \vee x_3)$ to an instance in special Subset-Sum.

the same vectors. Similarly, the vectors $\mathbf{z}_{j,1}, \cdots, \mathbf{z}_{j,m_j-1}$ are encoded as $g_{\mathbf{z}_{j,1}}^{\alpha^{\mathbf{z}_{j,1}}}$. The ciphertext for encrypting a message m is

$$\left\{g_{\mathbf{u}_{i,0}}^{\alpha^{\mathbf{u}_{i,0}}}, g_{\mathbf{u}_{i,1}}^{\alpha^{\mathbf{u}_{i,1}}}\right\}_{1\leq i\leq n} \cup \left\{g_{\mathbf{v}_{j,1}}^{\alpha^{\mathbf{v}_{j,1}}}, g_{\mathbf{z}_{j,1}}^{\alpha^{\mathbf{z}_{j,1}}}\right\}_{1\leq j\leq k} \cup \left\{m \cdot g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}\right\}$$

Hence the size of the ciphertext for the CNF formula is of $2n + 2k + 1$ group elements. The decryption involves $n + m - k$ mapping operations, that is the n maps to compute $g_{\sum_{i=1}^{n} \mathbf{u}_{i,x_i}}^{\alpha^{\sum_{i=1}^{n} \mathbf{u}_{i,x_i}}}$, and $m-k$ maps for $g_{\delta_j \mathbf{v}_{j,1}}^{\alpha^{\delta_j \mathbf{v}_{j,1}}}, g_{(m_j-\delta_j)\mathbf{z}_{j,1}}^{\alpha^{(m_j-\delta_j)\mathbf{z}_{j,1}}}$ for $1 \leq j \leq k$ which can be done in time $O\Big(\sum_{j=1}^{k} \big(\mathsf{log}(\delta_i) + \mathsf{log}(m_j - \delta_i)\big)\Big) \leq O\big(k \mathsf{log} \frac{m}{2k}\big)$. Hence the decryption has time complexity $n + O\big(k \mathsf{log} \frac{m}{2k}\big)$. The multilinearity level is $n + m - k$ and can be optimised to $n + \mathcal{O}\big(k \mathsf{log} \frac{m}{2k}\big)$ (see Appendix F for details).

3 Bitcoin protocol

In this section we first provide a short overview of the Bitcoin protocol. Then we discuss the probability of solving the proof-of-work puzzle used in Bitcoin. This gives a security basis for using Bitcoin to build a timed-release encryption.

3.1 Overview of Bitcoin protocol

Bitcoin [36] 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 are stored in a public distributed ledger called the *block-chain*.

The block-chain is an ordered, back-linked list of blocks of transactions, as shown in Figure 3. Each block is identified by a hash, generated using the $\mathsf{SHA256}$ cryptographic hash algorithm on the header of the block. The data structure of block header is given in Figure 2. Each block references a previous block through the "hashPreBlock" field in the block header. The fields of difficulty, timestamp, and nonce in Figure 2 are related to the mining competition. The merkle tree root is a data structure used to efficiently summarize all the transactions in the block.

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

Fig. 2: The structure of the block header

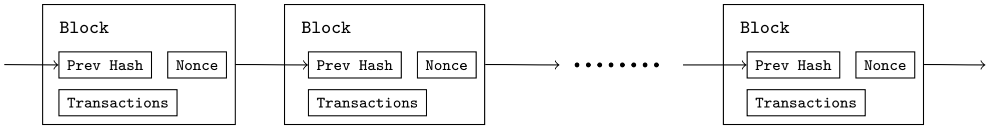

Fig. 3: Bitcoin block chain

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

$$\mathsf{SHA256}(\mathsf{SHA256}(\mathit{block\_header})) \leq 0^d \| 1^{256-d} \tag{1}$$

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 d is the number of leading zeroes which represents the difficulty of Bitcoin mining. 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.

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

Fig. 4: Proof-of-work constraints

3.2 Analysis of proof-of-work puzzle

We shall discuss the probability of finding a solution for the proof-of-work puzzle in Bitcoin mining. For simplicity, we analyse the case when the target is of the form $0^d 1^{256-d}$ with d a positive integer. The calculation of the actual target is more complex. The current target value is around $2^{188.53}$ (i.e., $d \approx 67$) in Aug 2015 [2]. Our method is also suitable for the case when the target is any other number.

Suppose the goal of Bitcoin mining is to find an auxiliary bit string $w \in \{0,1\}^{384}$ (including the nonce, transaction data etc., as described in the previous section) which hashes the hash of previous block-header $x \in \{0,1\}^{256}$ to a value under a target $\mathcal{E}_d$. Let $H : \{0,1\}^* \to \{0,1\}^{256}$ be the double hash used in the Bitcoin protocol. Assume the output of the hash function is uniformly distributed on $\{0,1\}^{256}$.

The following Proposition 1 gives the average number of trials for finding the first w such that $H(x,w) \leq \mathcal{E}_d$.

Proposition 1. *Let X be a random variable that denotes the number of trials for finding the first $w \in \{0,1\}^{384}$ such that $H(x,w) \leq \mathcal{E}_d$ where $x \in \{0,1\}^{256}$ and the target $\mathcal{E}_d = 0^d 1^{256-d}$ and H is a hash function whose output is uniformly distributed on $\{0,1\}^{256}$. Then the expectation $E[X] \approx 2^d$.*

Proof. See Appendix A.

The next proposition gives the probability of finding at least one solution w within n trials.

Proposition 2. *Let $P(d,n)$ be the probability of finding at least one w such that $H(x,w) \leq \mathcal{E}_d$ within n trials with $x, w, \mathcal{E}_d, H$ defined similar as in above proposition. Then*

1. $P(d,n) = 1 - \left(1 - \frac{1}{2^d}\right)^n$*;*

2. $1 - e^{-\frac{n}{2^d}} < P(d,n) < \frac{n}{2^d}$.

Proof. See Appendix A.

The following proposition gives an upper bound on the probability of computing ℓ continuous blocks within n trials in total.

Proposition 3. *Given a pair* (x_0, w_0) *such that* $H(x_0, w_0) \le \mathcal{E}_d$*, let* $P(d,n,\ell)$ *be the probability of finding* ℓ *continuous pairs* (x_i, w_i) *such that* $H(x_i, w_i) = x_{i+1}$ *and* $H(x_{i+1}, w_{i+1}) \le \mathcal{E}_d$ *within* n *trials in total. Then* $P(d,n,\ell) \le n^\ell/(\ell \cdot 2^d)^\ell$*.*

Proof. See Appendix A.

4 Timed-release encryption from Bitcoin and witness encryption

We design a new timed-release encryption by using Bitcoin block-chain as a public time clock. Bitcoin network is decentralised and there is no trusted third party involved in our protocol. We use Bitcoin proof-of-work constraints as a "public key", encrypt with witness encryption [25] and decrypt with any solution of the constraint.

4.1 Building blocks

In this section, we formally define the notion of *timed-release encryption* (TRE). Our TRE scheme consists of two building blocks: a *timer* for modelling the Bitcoin block-chain, and witness encryption. Below we give the primitives for modelling Bitcoin blockchain as a timer and then give the intuitive explanations:

Definition 4 (Timer). *A timer is a 3-tuple* $(T, \prec, \mathsf{Tick})$ *where* $T \subset \{0,1\}^*$ *is a time-point space,* $\prec$ *is a NP relation on* T*, and* Tick *is a probabilistic algorithm that takes a time-point* $\tau \in T$ *as input and outputs* τ' *with* $\tau \prec \tau'$ *or an error* $\perp$*. We denote by* $\tau \prec^k \tau'$ *for* $k \ge 1$ *when there are* $\tau_1, \cdots, \tau_k$ *such that* $\tau \prec \tau_1 \prec \tau_2 \prec \cdots \prec \tau_k = \tau'$*. We require a timer to be live:*

- **Liveness:** *there exists a negligible function* $\mu(\cdot)$ *such that for any* $t \in T$*,*

$$Pr[\tau' \leftarrow \mathsf{Tick}(\tau) : \tau \prec \tau'] \ge 1 - \mu(|\tau|)$$

By using the notation in Section 3.2, when the timer is instantiated with bitcoin protocol, a time-point τ can be a pair (x, w), and $\tau \prec \tau'$ when $\tau' = (H(x,w), w')$ and $H(H(x,w), w') \le \mathcal{E}_d$. The liveness states that the Tick algorithm proceeds to the next time-point with a overwhelming probability. The timer might not be able to continue with negligible probability. This is the exact case for the bitcoin mining procedure as we analysed in Proposition 2.

The goal of timed-release encryption is to delay the decryption for a certain period of time. It is computationally feasible for the adversary to decrypt as long as it puts in enough computational work. Hence the concrete security is more suitable than the asymptotic security.

We define a *time-oracle* $\mathcal{T}(\tau, \cdot)$ for $\tau \in T$ on a timer $(T, \prec, \mathsf{Tick})$. The oracle creates an internal list L and initialises it as $L := \{\tau\}$. The adversary can make the *update* queries and the oracle responses as follows: the oracle retrieves the last entry in the list, say τ, and runs $\tau' \leftarrow \mathsf{Tick}(\tau)$. The oracle gives τ' to the adversary. If $\tau \prec \tau'$ (which implies $\tau' \neq \perp$), then the oracle appends τ' to the list.

Definition 5 **($(t, q, \ell; \varepsilon)$-security for the timer).** *A timer $(T, \prec, \mathsf{Tick})$ is $(t, q, \ell; \varepsilon)$-secure if for each adversary $\mathcal{A}$ that runs in time t, query time-oracle at most q times with $q < \ell$, for any $\tau \in T$, it holds that*

$$Pr\Big[(\tau_1, \cdots, \tau_\ell) \leftarrow \mathcal{A}^{\mathcal{T}(\tau, \cdot)}(\tau, \ell) : \tau \prec \tau_1 \prec \cdots \prec \tau_\ell\Big] \leq \varepsilon$$

We allow the adversary to query the oracle for at most q ($q < \ell$) times. If the adversary queries the oracle for ℓ times, then it becomes trivial since it means the adversary honestly waits for the sufficient amount of time to pass and wins with probability 1. The adversary tries to compute a chain of time-points of length ℓ starting from τ. The winning probability depends on the length ℓ of the required time-points, as demonstrated in Proposition 3. The more computational work the adversary put in, the bigger probability the adversary find the solution. In the bitcoin network, the Bitcoin peers, i.e., the “adversaries”, compete to compute the proof-of-work to generate the next block and the fastest “adversary” wins the bonus.

Definition 6 **(Timed-release encryption (TRE)).** *A timed-release encryption* TRE *on a timer $(T, \prec, \mathsf{Tick})$ consists of two polynomial-time algorithms:*

- $\mathsf{TRE.Enc}\big(1^\lambda, (\tau, \ell), m\big)$ *is an encryption algorithm that takes as input a starting time-point $\tau \in T$ and an integer ℓ, and a message $m \in \mathbb{M}$, and outputs a ciphertext c.*
- $\mathsf{TRE.Dec}(c, k)$ *is a decryption algorithm that takes as input a ciphertext c and a string $k \in \{0,1\}^*$, and outputs $m \in \mathbb{M} \cup \{\perp\}$.*
- **Correctness.** *If $\tau \prec \tau_1 \prec \cdots \prec \tau_\ell$ then*

$$Pr\Big[\mathsf{TRE.Dec}\Big(\mathsf{TRE.Enc}\big(1^\lambda, (\tau, \ell), m\big), \{\tau_j\}_{j=1}^{\ell}\Big) = m\Big] = 1$$

The timed-release encryption is defined on a timer. The ciphertext is encrypted to a starting point τ for timing, and the expected decryption time is after the ℓ ticks of the timer. The correctness states that the algorithm can decrypt when the ℓ continuous time-points starting from τ are found.

Definition 7 **($(t, q, \ell; \varepsilon)$-security for TRE).** *A timed-release encryption on a timer $(T, \prec, \mathsf{Tick})$ is $(t, q, \ell; \varepsilon)$-secure if for every adversary $\mathcal{A}$ run in time t and* *makes at most q ($q < \ell$) times update queries, for any $\tau \in T$ it holds that*

$$Pr\left[\begin{array}{l} (m_0, m_1, St) \leftarrow \mathcal{A}_0^{\mathcal{T}(\tau,\cdot)}\big(1^\lambda, \tau, \ell\big) \\ b \leftarrow \{0,1\}, c \leftarrow \mathsf{TRE.Enc}\big(1^\lambda, (\tau,\ell), m_b\big) \; : b = b' \\ b' \leftarrow \mathcal{A}_1^{\mathcal{T}(\tau,\cdot)}\big(1^\lambda, \tau, \ell, m_0, m_1, c, St\big) \end{array}\right] \leq \varepsilon$$

The above security definition measures the probability that the adversary can decrypt within time t. The more computational work the adversary puts in, the bigger probability that he can decrypt with.

Construction 2 (Timed-release encryption) *We now present the construction of timed-release encryption. What we do is we encrypt our messages using a witness encryption scheme relative to the NP relation for our timer. The witness would be a valid chain of ℓ time points.*

$$\begin{aligned} &\mathsf{TRE.Enc}\big(1^\lambda, (\tau,\ell), m\big) := \mathsf{WE.Enc}\big(1^\lambda, x, m\big) \text{ with } x := (\tau, \ell) \\ &\mathsf{TRE.Dec}(c, k) := \mathsf{WE.Dec}(c, k) \end{aligned}$$

Based on the security of the timer and the witness encryption, we can prove the security of the time-release encryption:

Theorem 3. *Given access to a time-oracle, our construction for timed-release encryption is $(t, q, \ell; \varepsilon)$-secure if the witness encryption is $(t, q, t', q'; \varepsilon, \varepsilon')$-secure and the timer is $(t', q', \ell; \varepsilon')$-secure.*

Proof. For a timed-release encryption adversary $\mathcal{A}$ that breaks the $(t, q, \ell; \varepsilon)$-security of encryption on (τ, ℓ) and has access to a time-oracle $\mathcal{T}(\tau, \cdot)$, it gives a witness encryption adversary that runs in time t and has access to the oracle $O(x, \cdot) := \mathcal{T}(\tau, \cdot)$ with $x = (\tau, \ell)$ and guesses b correctly with probability more than ε. If the witness encryption is $(t, q, t', q'; \varepsilon, \varepsilon')$-secure, this gives an extractor that runs in time t' and queries the oracle for at most q' times and extracts a witness with probability more than ε'. Recall that this witness is a continuous chain of time-points into the future, which contradicts the assumption that the timer is $(t', \ell; \varepsilon')$-secure.

5 Conclusion

We have proposed a new witness encryption based on Subset-Sum which achieves extractable security without relying on obfuscation and is more efficient than the existing ones. Our witness encryption employs multilinear maps of arbitrary order and it is independent of the implementations of multilinear maps. For encoding a CNF formula of n variables and k clauses, our witness encryption achieves ciphertext of $2n+2k+1$ group elements, as compared with other known approaches for which the size is $\mathsf{poly}(n, k)$.

As an application, we construct a new timed-release encryption based on the Bitcoin protocol and our extractable 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 novelty of our protocol is that the decryption key will be automatically revealed in the bitcoin block-chain when the block-chain reaches a certain length.

Acknowledgments

We are grateful to Mark Ryan and Flavio Garcia for enlightening discussions and collaboration at an early stage of this research.

References

1. Bitcoin Block Explorer. `https://blockexplorer.com`.
2. Bitcoin target. http://blockexplorer.com/q/hextarget, Aug, 2015.
3. M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Fair Two-Party Computations via Bitcoin Deposits. In *FC*, pages 105–121, 2014.
4. M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure Multiparty Computations on Bitcoin. SP '14, pages 443–458, 2014.
5. B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai. Protecting Obfuscation against Algebraic Attacks. In *EUROCRYPT*, pages 221–238, 2014.
6. M. Bellare and S. Goldwasser. Verifiable partial key escrow. In *CCS*, 1997.
7. M. Bellare and V. T. Hoang. Adaptive witness encryption and asymmetric password-based cryptography. In *PKC*, pages 308–331, 2015.
8. I. Bentov and R. Kumaresan. How to Use Bitcoin to Design Fair Protocols. In *CRYPTO*, pages 421–439, 2014.
9. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. CRYPTO '01, pages 213–229. Springer-Verlag, 2001.
10. D. Boneh and M. Naor. Timed commitments. In *CRYPTO*, pages 236–254, 2000.
11. D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002.
12. D. Boneh, B. Waters, and M. Zhandry. Low Overhead Broadcast Encryption from Multilinear Maps. In *CRYPTO*, pages 206–223, 2014.
13. J. Bonneau, J. Clark, and S. Goldfeder. On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015, 2015.
14. Z. Brakerski and G. N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. *IACR Cryptology ePrint Archive*, 2013:563, 2013.
15. J. Cathalo, B. Libert, and J. Quisquater. Efficient and non-interactive timed-release encryption. In *ICICS*, pages 291–303, 2005.
16. 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.
17. J. H. Cheon, C. Lee, and H. Ryu. Cryptanalysis of the new clt multilinear maps. Cryptology ePrint Archive, Report 2015/934, 2015. `http://eprint.iacr.org/`.
18. E. Clarke, D. Kroening, and F. Lerda. A tool for checking ANSI-C programs. In *TACAS*, pages 168–176, 2004.
19. J. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, pages 476–493, 2013.
20. J. Coron, T. Lepoint, and M. Tibouchi. New multilinear maps over the integers. In *CRYPTO*, pages 267–286, 2015.

21. G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. EUROCRYPT'99, pages 74–89, 1999.
22. J. A. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In *EUROCRYPT*, pages 281–310, 2015.
23. S. Garg, C. Gentry, and S. Halevi. Candidate Multilinear Maps from Ideal Lattices. In *EUROCRYPT*, pages 1–17, 2013.
24. S. Garg, C. Gentry, S. Halevi, and D. Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, pages 518–535, 2014.
25. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness Encryption and Its Applications. STOC '13, pages 467–476, 2013.
26. C. Gentry, A. B. Lewko, and B. Waters. Witness Encryption from Instance Independent Assumptions. In *CRYPTO*, pages 426–443, 2014.
27. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO*, pages 536–553, 2013.
28. T. Jager. How to build time-lock encryption. Cryptology ePrint Archive, Report 2015/478, 2015.
29. 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`.
30. R. Kumaresan, T. Moran, and I. Bentov. How to Use Bitcoin to Play Decentralized Poker. In *CCS*, pages 195–206, 2015.
31. A. Langlois, D. Stehlé, and R. Steinfeld. GGHLite: More Efficient Multilinear Maps from Ideal Lattices. In *EUROCRYPT*, pages 239–256, 2014.
32. A. Miller.
33. Norbert Manthey. Riss 4.27 (with Coprocessor, Qprocessor, Mprocessor, ShiftBMC, Priss, Pcasso and BlackBox). `http://tools.computational-logic.org/content/riss427.php`.
34. K. G. Paterson and E. A. Quaglia. Time-specific encryption. In *SCN*, pages 1–16, 2010.
35. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, 1996. MIT/LCS/TR-684.
36. Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`.
37. Timothy May. Timed-release crypto. manuscript, February 1993.
38. Yupu Hu and Huiwen Jia. Cryptanalysis of GGH Map. `https://eprint.iacr.org/2015/301.pdf`.
39. M. Zhandry. How to Avoid Obfuscation Using Witness PRFs. In *Proceedings of TCC*, 2016.
40. Z. Zhao and T.-H. H. Chan. How to vote privately using bitcoin. Cryptology ePrint Archive, Report 2015/1007, 2015.
41. J. Zimmerman. How to Obfuscate Programs Directly. In *EUROCRYPT Part II*, pages 439–467, 2015.

APPENDIX

A Proofs in Section 3

Proposition 1. Let X be a random variable that denotes the number of trials for finding the first $w \in \{0,1\}^{384}$ such that $H(x,w) \leq \mathcal{E}_d$ where $x \in \{0,1\}^{256}$ and the target $\mathcal{E}_d = 0^d 1^{256-d}$ and H is a hash function whose output is uniformly distributed on $\{0,1\}^{256}$. Then the expectation $E[X] \approx 2^d$.

Proof. Let $Pr[X = j]$ be the probability of finding the first w at the j-th attempts. For each attempt w, the probability $Pr[H(x,w) \leq \mathcal{E}] = 2^{256-d}/2^{256} = 1/2^d$. Then

$$
\begin{aligned}
E[X] &= \sum_{j=1}^{2^{384}} jP(X=j) = \sum_{j=1}^{2^{384}} P(X \geq j) = \sum_{j=1}^{2^{384}} \left(1 - \frac{1}{2^d}\right)^{j-1} \\
&= 2^d \left(1 - \left(1 - \frac{1}{2^d}\right)^{2^{384}}\right)
\end{aligned}
$$

Since $\lim_{m\to\infty}(1 - \frac{1}{m})^m = 1/e$ and d is at most 256, we have $(1 - 1/2^d)^{2^{384}} \leq (1 - 1/2^{256})^{2^{384}} \approx (1/e)^{2^{128}} \approx 0$. Hence $E[X] \approx 2^d$.

Proposition 2. Let $P(d,n)$ be the probability of finding at least one w such that $H(x,w) \leq \mathcal{E}_d$ within n trials with $x, w, \mathcal{E}_d, H$ defined similar as in above proposition. Then

1. $P(d,n) = 1 - \left(1 - \frac{1}{2^d}\right)^n$;
2. $1 - e^{-\frac{n}{2^d}} < P(d,n) < \frac{n}{2^d}$.

Proof. we shall use the upper bound of the series $(1 - \frac{1}{m})^m$ for $m \geq 1$. Since $e^{-x} \geq 1 - x$ for $x \in (0,1]$, we have $1/e \geq (1-x)^{1/x}$. Replacing x with $1/m$, we get $1/e \geq (1 - 1/m)^m$ for any $m \geq 1$.

1. The probability of finding no solution in n trials is $(1 - 1/2^d)^n$ as shown in Proposition 1. Hence the probability of finding at least one solution is $P(d,n) = 1 - \left(1 - \frac{1}{2^d}\right)^n$.

2. - For the upper bound, clearly sampling with replacement gives smaller probability compared to sampling without replacement:

$$\begin{aligned} P(d,n) &= 1 - \left(1 - \frac{1}{2^d}\right)^n \\ &< 1 - \left(1 - \frac{1}{2^d}\right)\left(1 - \frac{1}{2^d - 1}\right) \cdots \left(1 - \frac{1}{2^d - n + 1}\right) \\ &= 1 - \frac{2^d - n}{2^d} = \frac{n}{2^d} \end{aligned}$$

   - For the lower bound, since $\left(1 - 1/2^d\right)^n = \left(\left(1 - 1/2^d\right)^{2^d}\right)^{n/2^d} \leq e^{-n/2^d}$, we have $P(d,n) \geq 1 - e^{-n/2^d}$.

Proposition 3. Given a pair (x_0, w_0) such that $H(x_0, w_0) \leq \mathcal{E}_d$, let $P(d,n,\ell)$ be the probability of finding ℓ continuous pairs (x_i, w_i) such that $H(x_i, w_i) = x_{i+1}$ and $H(x_{i+1}, w_{i+1}) \leq \mathcal{E}_d$ within n trials in total. Then $P(d,n,\ell) \leq n^\ell/(\ell \cdot 2^d)^\ell$.

Proof. Since we assume the output of the hash function is uniformly distributed, the probability of finding each following pair is independent of each other. Hence $P(d,n,\ell) = P(d,n_1) \cdots P(d,n_\ell)$ with n_j the number of trials used for computing the j-th pair and $n_1 + n_2 + \cdots + n_\ell = n$.

$$\begin{aligned} P(d,n,\ell) &= \left(1 - \left(1 - \frac{1}{2^d}\right)^{n_1}\right) \cdots \left(1 - \left(1 - \frac{1}{2^d}\right)^{n_\ell}\right) \\ &\leq \frac{n_1}{2^d} \cdots \frac{n_\ell}{2^d} \leq \left(\frac{n}{\ell}\right)^\ell \cdot \left(\frac{1}{2^d}\right)^\ell = \left(\frac{n}{\ell \cdot 2^d}\right)^\ell \end{aligned}$$

The first inequality is by Proposition 2 and the second inequality is because of $x_1 \cdots x_\ell \leq \left(\frac{x_1 + \cdots + x_\ell}{\ell}\right)^\ell$.

B Generic multilinear map model

We will make use of asymmetric multilinear maps in which groups are indexed by integer vectors [23, 12, 41].

Given a ring R and a universe multi-set U, an element is $[x]_S$ where $x \in R$ is the value of the element and $S \subseteq U$ is the index of the element. The binary operations over elements are defined as follows:

- For two elements $e_1 := [x_1]_S$ and $e_2 := [x_2]_T$ such that $S = T$, $e_1 + e_2$ is defined as $[x_1 + x_2]_S$, and $e_1 - e_2$ is defined as $[x_1 - x_2]_S$.
- For two elements $e_1 := [x_1]_S$ and $e_2 := [x_2]_T$, $e_1 \cdot e_2$ is defined as $(x_1 x_2, S \uplus T)$ for $S \uplus T \subseteq U$ where $\uplus$ is the multi-set union.

A *generic multilinear map oracle* [5,41] is a stateful oracle $\mathcal{M}$ that responds to queries as follows:

- **Initialization.** $\mathcal{M}$ will be initialized with a ring R, a universe set U, and a list L of initial elements. For every $[x]_S \in L$, $\mathcal{M}$ generates a handle h. The value of the handles are chosen to be independent of the elements being encoded, and the handles are distinct. $\mathcal{M}$ outputs the handles generated for all the elements in L. After the initialisation, all subsequent calls to initialization queries fail.
- **Algebraic operations.** Given two input handles h_1, h_2 and an operation $\circ \in (+, -, \cdot)$, $\mathcal{M}$ first locates the relevant elements e_1, e_2 in the handle table. If any of the input handles does not appear in the handle table, the call to $\mathcal{M}$ fails. Then $\mathcal{M}$ computes $e_1 \circ e_2$; the call fails if $e_1 \circ e_2$ is undefined. $\mathcal{M}$ generates a new handle for $e_1 \circ e_2$, saves this element and the new handle in the handle table, and returns the new handle.
- **Zero-test.** Given an input handle h, $\mathcal{M}$ first locates the relevant element $[x]_S$ in the handle table. If h does not appear in the handle table, the call fails. If $S \neq U$ the call fails. Then $\mathcal{M}$ returns 1 if $x = 0$, otherwise returns 0.

The zero-test is only available on the top-level index. The zeros on different index levels are different.

C Extractability Security

Theorem 1. Our Construction 1 of witness encryption achieves extractable security with $t' = \mathsf{poly}(t \cdot \lambda)$ and $\varepsilon' = 2\varepsilon - 1$ and $q' \leq q$ in the generic model of multilinear maps.

Proof of Theorem 1 Assume an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that run in time t, and an instance $x \in \{0,1\}^*$

$$Pr\left[\begin{array}{l}(m_0, m_1, St) \leftarrow \mathcal{A}_0^{O(x,\cdot)}\big(1^\lambda, x\big) \\ b \leftarrow \{0,1\}\,; c \leftarrow \mathsf{WE.Enc}\big(1^\lambda, x, m_b\big) \;: b = b' \\ b' \leftarrow \mathcal{A}_1^{O(x,\cdot)}\big(1^\lambda, x, m_0, m_1, c, St\big)\end{array}\right] > \varepsilon$$

Then we shall prove there exists an extractor E that run in time $\Theta(t \cdot \lambda^2)$ such that

$$Pr\Big[w \leftarrow E^{O(x,\cdot)}\big(1^\lambda, x\big) : (x, w) \in R_L\Big] > 2\varepsilon - 1$$

To ease the discussion, we consider the instances of the special Subset-Sum problem defined in Definition 9, rather than the original SAT problem.

We construct an extractor E by using $\mathcal{A}$ as subroutine to extract a witness. E queries to the oracle $O(x, \cdot)$ and forwards the responses to the adversary whenever the adversary queries to $O(x, \cdot)$. E gives x to $\mathcal{A}$. $\mathcal{A}$ chooses (m_0, m_1) and sends them to E. Let x be the instance of special Subset-Sum: given a multi-set of d-vectors $\mathit{\Delta} = \{(\mathbf{v}_i : \ell_i)\}_{i\in I}$ of positive integers and a target sum vector $\mathbf{s}$ of positive integers such that $(\ell_i + 1)\mathbf{v}_i \not\leq \mathbf{s}$ for each $i \in I$.

E chooses $b \leftarrow (0, 1)$ uniformly at random. E encodes the instance x according to our witness encryption scheme as $\left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I} \cup \left\{g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}\right\}$ where α is chosen uniformly at random. E chooses d fresh formal symbols $\mathcal{X} = (X_1, \cdots, X_d)$ and creates an initial list $L := \left\{h_i \mapsto [y_i]_{V_i}\right\}_{i\in I} \cup \{h \mapsto [y]_T, h' \mapsto [y']_T, h'' \mapsto [y'']_T\}$ where

- $V_i := \mathcal{X}^{\mathbf{v}_i}$ and $T := \mathcal{X}^{\mathbf{s}}$.
- The handles h_i, h are chosen uniformly at random.
- y_i, y are fresh variables.

Intuitively, y_i represents $\alpha^{\mathbf{v}_i}$, y represents $m_b + \alpha^{\mathbf{s}}$, y' represents m_0 and y'' represents m_1. Those variables are used to keep track of adversary's computation. E gives those handles $\{h_i\}_{i\in I} \cup \{h, h', h''\}$ to $\mathcal{A}$. E answers the multilinear maps oracle queries for addition, subtraction and multiplication as defined in Appendix B. To answer the zero-test queries, E instantiates those variables with its real values and test whether the results are zeros.

Note that our proof also works when the handles for group generators $\widehat{h} \mapsto [1]_T$ and $\widehat{h}_i \mapsto [1]_{V_i})$ for $i \in I$ are given to the adversary, although our scheme itself does not need to give the group generators. The elements of the other index level are hidden.

We define an extraction function ext over the polynomials z produced by the adversary. The $\mathsf{ext}(z)$ returns an $|I|$-vector of integers as follows:

$$\begin{aligned}
&\text{For } i \in I, \mathsf{ext}(y_i) = \mathbf{u} \text{ with } \mathbf{u} \text{ a } |I|\text{-vector of 0 except a 1 on its } i\text{-th element}\\
&\mathsf{ext}(y) = \mathsf{ext}(y') = \mathsf{ext}(y'') = \mathbf{0}\\
&\mathsf{ext}(z_1 \cdot z_2) = \mathsf{ext}(z_1) + \mathsf{ext}(z_2)\\
&\mathsf{ext}(z_1 + z_2) = \begin{cases} \mathsf{ext}(z_1) & \text{if } \mathsf{ext}(z_1) \neq \mathbf{0}\\ \mathsf{ext}(z_2) & \text{otherwise}\end{cases}\\
&\mathsf{ext}(z_1 - z_2) = \begin{cases} \mathsf{ext}(z_1) & \text{if } \mathsf{ext}(z_1) \neq \mathbf{0}\\ \mathsf{ext}(z_2) & \text{otherwise}\end{cases}
\end{aligned}$$

Apparently this extraction function can be efficiently computed for any polynomial z.

Lemma 1. *For any formal polynomials z produced by $\mathcal{A}$ at an index level $U := \mathcal{X}^{\mathbf{u}}$ with $U \subseteq T$ for some d-vector $\mathbf{u}$ ($\mathbf{u} \leq \mathbf{s}$). Let $\mathsf{ext}(z) = \left(\delta_1, \cdots, \delta_{|I|}\right)$.*

1. *$\mathsf{ext}(z) = \mathbf{0}$ iff z is constructed by only using variable y, y', y''.*
2. *If $\mathsf{ext}(z) \neq \mathbf{0}$, then $\sum_{i\in I} \delta_i \mathbf{v}_i = \mathbf{u}$ and $0 \leq \delta_i \leq \ell_i$ for each $i \in I$.*

Proof. The proof of the two results goes by induction on the structure of z.

1. if $z = y_i$, its index level is $\mathcal{X}^{\mathbf{v}_i}$ and $\mathsf{ext}(z) = (\delta_1, \cdots, \delta_{|I|})$ with $\delta_i = 1$ and $\delta_j = 0$ for $j \neq i$. Clearly we have $\sum_{j\in I} \delta_j \mathbf{v}_j = \mathbf{v}_i$.
2. if z is y, y', or y'', we have $\mathsf{ext}(z) = \mathbf{0}$.

3. if $z = z_1 + z_2$, according to the definition of $\mathcal{M}$, we know z_1, z_2 are both the polynomials of the same index level $\mathcal{X}^{\mathbf{u}}$. By induction hypothesis, $\mathsf{ext}(z_1) = \mathbf{0}$ ($\mathsf{ext}(z_2) = \mathbf{0}$) iff z_1 (z_2) is constructed by only using y. By definition of ext, for $\mathsf{ext}(z)$ to be $\mathbf{0}$, it must be $\mathsf{ext}(z_1)$ and $\mathsf{ext}(z_2)$ are both $\mathbf{0}$, i.e., both z_1 and z_2 are constructed by only using y, y', y''. That is to say $\mathsf{ext}(z) = \mathbf{0}$ iff z is constructed by only using y, y', y''. Hence the first result holds.
   W.l.o.g., assume $\mathsf{ext}(z_1) \neq \mathbf{0}$. Let $\mathsf{ext}(z_1) = \big(\delta_1, \cdots, \delta_{|I|}\big)$. By induction hypothesis, we have $\sum_{i \in I} \delta_i \mathbf{v}_i = \mathbf{u}$, and $\delta_i \leq \ell_i$ for each $i \in I$. By definition, $\mathsf{ext}(z) = (\delta_1, \cdots, \delta_{|I|})$, and hence the second result holds.
4. if $z = z_1 - z_2$, simiar as above.
5. if $z = z_1 \cdot z_2$, by definition of ext, for $\mathsf{ext}(z)$ to be $\mathbf{0}$, it must be $\mathsf{ext}(z_1)$ and $\mathsf{ext}(z_2)$ are both $\mathbf{0}$, i.e., both z_1 and z_2 are constructed by only using y, y', y''. That is to say $\mathsf{ext}(z) = \mathbf{0}$ iff z is constructed by only using y, y', y''. Hence the first result holds.
   Let $\mathsf{ext}(z_1) = \left(\delta'_1, \cdots, \delta'_{|I|}\right)$ and $\mathsf{ext}(z_2) = \left(\delta''_1, \cdots, \delta''_{|I|}\right)$. Assume z_1 is a polynomial on index level $\mathcal{X}^{\mathbf{u}}$ and z_2 is a polynomial on index level $\mathcal{X}^{\mathbf{w}}$. Then z is a polynomial on index level $\mathcal{X}^{\mathbf{u}+\mathbf{w}}$. By induction hypothesis, we know that $\sum_{i \in I} \delta'_i \mathbf{v}_i = \mathbf{u}$ and $\delta'_i \leq \ell_i$ for each $i \in I$, and $\sum_{i \in I} \delta''_i \mathbf{v}_i = \mathbf{w}$ and $\delta''_i \leq \ell_i$ for each $i \in I$. Since $\mathsf{ext}(z) = \left(\delta'_1 + \delta''_1, \cdots, \delta'_{|I|} + \delta''_{|I|}\right)$, we have $\sum_{i \in I} (\delta'_i + \delta''_i) \mathbf{v}_i = \mathbf{u} + \mathbf{w}$.
   We are now left to show that for any $i \in I$, $\delta'_i + \delta''_i \leq \ell_i$. Assume for some $i \in I$, $\delta'_i + \delta''_i > \ell_i$. We shall show that $U \not\subseteq T$. From the condition $\ell_i \mathbf{v}_i \not\leq \mathbf{s}$, we know that $(u_1, \cdots, u_d) \not\leq \mathbf{s}$. But this multiplication will be rejected by the multilinear map oracle. This contradicts the assumption.

Lemma 2. *A formal polynomial z produced by the adversary $\mathcal{A}$ on the top index level $\mathbf{s}$ can be rewritten into $z = f(y_1, \cdots, y_{|I|}) + a \cdot y + a' \cdot y' + a'' \cdot y''$ for some integers a, a', a'' by only collecting the like terms of y, y', y''. Moreover the polynomial $f(y_1, \cdots, y_{|I|})$ is on the top index level and does not contain y, y', y'' and can also be computed by the adversary.*

Proof. The initial top index level elements $h \mapsto [y]_T, h' \mapsto [y']_T, h'' \mapsto [y'']_T$ can only be added and subtracted. No multiplication is allowed on the top level. Even if the polynomial only contains $z' - z'$ with z' on the top index level, this polynomial cannot be multiplied since $z' - z'$ is a zero on the top index level. That is to say y, y', y'' do not occur inside of any multiplication operations. Hence we can separate y, y', y'' by using communicative and associative rules on $+$. After collecting the like terms of y (no expansion or cancellation of the other terms), z can be rewritten as $z = f(y_1, \cdots, y_{|I|}) + ay + a'y' + a''y''$ for some polynomial f (which does not contain y, y', y'') and some integers a, a', a''. Clearly for any polynomial produced by $\mathcal{A}$, its communicative and associative equivalence can also be computed efficiently by $\mathcal{A}$ by simply changing the order of addition and subtraction on the top level.

Let Good be an event that the adversary can construct such a $z = f(y_1, \cdots, y_{|I|}) + ay + a'y' + a''y''$ with $f(y_1, \cdots, y_{|I|}) \neq 0$ (note that we didn't cancel terms in f. So $f(y_1, \cdots, y_{|I|})$ can only be 0 when it does not exist at all. Even if f only contains terms like $z' - z'$, f is not 0). If Good occurs, by definition $\mathsf{ext}\big(f(y_1, \cdots, y_{|I|})\big) \neq \mathbf{0}$. Assume $\mathsf{ext}\big(f(y_1, \cdots, y_{|I|})\big) = (\delta_1, \cdots, \delta_{|I|})$, then from Lemma 1, we have $\sum_{i \in I} \delta_i \mathbf{v}_i = \mathbf{s}$ and $0 \leq \delta_i \leq \ell_i$ for $i \in I$. In other words, we get a subset sum for $\mathbf{s}$. If Good does not occur, this means the adversary is not able to construct any polynomial that contains elements from both $\{y_1, \cdots, y_{|I|}\}$ and $\{y, y', y''\}$, hence the adversary can only constructs the polynomial of the form $f_1\big(y_1, \cdots, y_{|I|}\big)$ and $f_2(y, y', y'')$. Recall that y represents $m_b + \alpha^{\mathbf{s}}$, y' represents m_0 and y'' represents m_1. In any polynomial $f_2(m_b + \alpha^{\mathbf{s}}, m_0, m_1)$, the distribution of $m_b + \alpha^{\mathbf{s}}$ is exactly the same as any uniform random in the ring because α is chosen uniformly at random. Hence the adversary can only guess the value of b and we have

$$\begin{aligned} Pr[b = b'] = Pr[b = b' \wedge \mathsf{Good}] + Pr[b = b' \wedge \neg\mathsf{Good}] \leq Pr[\mathsf{Good}] + \frac{1}{2} Pr[\neg\mathsf{Good}] \\ = \frac{1}{2} + \frac{1}{2} Pr[\mathsf{Good}] \end{aligned}$$

Since $Pr[b = b'] > \varepsilon$, we have $Pr[\mathsf{Good}] > 2\varepsilon - 1$. And the event Good is in fact the event the extractor can extract a witness for the instance of Subset-Sum.

The extractor initialises the multilinear map oracle and it takes time $\mathsf{poly}(\lambda)$. The extractor performs the group operations of addition, subtraction, multiplication and zero-testing queried by the adversary and it takes $t \cdot \mathsf{poly}(\lambda)$ time since the addition, subtraction and multiplication take time $\mathsf{poly}(\lambda)$ and zero-testing is in time $t \cdot \mathsf{poly}(\lambda)$ since polynomial produced by the adversary is of depth at most $\Theta(t)$. The extraction algorithm is in time $\Theta(t)$. Hence it takes the extractor $\mathsf{poly}(t \cdot \lambda)$ times in total. This concludes the proof.

D Soundness security

Definition 8 (Soundness security [25] for WE). *A witness encryption scheme for a NP language L is (t, ε)-secure if for any adversary $\mathcal{A}$ that runs in time t, for any $x \notin L$, we have:*

$$\big|Pr\big[\mathcal{A}\big(\mathsf{WE.Enc}(1^\lambda, x, 0)\big) = 1\big] - Pr\big[\mathcal{A}\big(\mathsf{WE.Enc}(1^\lambda, x, 1)\big) = 1\big]\big| \leq \varepsilon$$

The soundness security states that if $x \notin L$ then no adversary that runs in time t can break the scheme with probability more than ε. An alternative definition for soundness security called adaptive soundness is given in [7].

The soundness security of our scheme will be based on the following *multilinear counting subset-sum Diffie-Hellman (mCSDH) assumption*:

Definition 9 ((t, ε)-secure mCSDH Assumption). *Given a multi-set of d-vectors $\Delta = \{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ of positive integers and a sum d-vector $\mathbf{s}$ of positive integers such that $(\ell_i + 1)\mathbf{v}_i \nleq \mathbf{s}$ for each $i \in I$.*

Let $\mathsf{param} \leftarrow \mathcal{G}\big(1^\lambda, \varDelta, \mathbf{s}\big)$ *be a description of a multilinear group family 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* $\mathbf{u}+\mathbf{v} \leq \mathbf{s}$*, together with group generators* $\{g_{\mathbf{v}}\}_{\mathbf{v}\leq\mathbf{s}}$*. Choose a vector of randoms* $\alpha := \langle \alpha_1, \cdots, \alpha_d \rangle$ *and a random* r*. If* $\mathbf{s}$ *cannot be represented as a subset-sum of elements from set* $\varDelta$*, then for any distinguisher* D *that runs in time* t*,*

$$\left| Pr\left[D\left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}\right) = 1\right] - Pr\left[D\left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, g_{\mathbf{s}}^{r}\right) = 1\right]\right| \leq \varepsilon$$

Theorem 4. *Our Construction 1 for witness encryption is* (t,ε)*-secure scheme under* (t',ε)*-mCSDH assumption where* $t' = t + \mathsf{poly}(\lambda)$ *where* $\mathsf{poly}(\lambda)$ *is the time for setting up the game and computing the challenge ciphertext.*

Proof. Let x be an instance of our special Subset-Sum: given a multi-set of d-vectors $\varDelta = \{(\mathbf{v}_i : \ell_i)\}_{i\in I}$ of positive integers and a target sum d-vector $\mathbf{s}$ of positive integers such that $\ell_i \mathbf{v}_i \not\leq \mathbf{s}$ for each $i \in I$.

Suppose an adversary $\mathcal{A}$ breaks the (t,ε)-security on x, then we can construct a distingusher to break Assumption 9. Let $\mathsf{param} \leftarrow \mathcal{G}\big(1^\lambda, \varDelta, \mathbf{s}\big)$ be a description of a multilinear group family. Choose a vector of randoms $\alpha := \langle \alpha_1, \cdots, \alpha_d \rangle$. D is given $U_0 := \left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}\right)$ or $U_1 := \left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, g_{\mathbf{s}}^{r}\right)$ as input. We denote this input by $\left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, K\right)$ where $K = g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}$ or $g_{\mathbf{s}}^{r}$. D chooses $b \leftarrow \{0,1\}$ and encrypts as $c_b = \left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, g_{\mathbf{s}}^{b} \cdot K\right)$. D gives c_b to $\mathcal{A}$. $\mathcal{A}$ outputs its guess b' for b. If $b = b'$ then D outputs 1; otherwise outputs 0. When D gets U_1, K is uniformly at random and hence c contains no information about b. In this case, $\mathcal{A}$ can only guess. That is $Pr[D(U_1) = 1] = 1/2$. When D gets U_0, from $\mathcal{A}$'s view, $\mathcal{A}$ is playing the perfect soundness security game. Hence $Pr[D(U_0) = 1] = \frac{1}{2} \cdot Pr[\mathcal{A}(c_0) = 0] + \frac{1}{2} \cdot Pr[\mathcal{A}(c_1) = 1] = \frac{1}{2} - \frac{1}{2} \cdot (Pr[\mathcal{A}(c_0) = 1] - Pr[\mathcal{A}(c_1) = 1])$. We have $|Pr[D(U_0) = 1] - Pr[D(U_1) = 1]| = \frac{1}{2}|Pr[\mathcal{A}(c_0) = 1] - Pr[\mathcal{A}(c_1) = 1]|$. Hence if $\mathcal{A}$ breaks the (t,ε)-security, then D breaks the (t',ε)-mCSDH assumption, where $t' = t + \mathsf{poly}(\lambda)$ and $\mathsf{poly}(\lambda)$ denotes a constant number of steps used for computing the challenge ciphertext.

Theorem 5. *The mCSDH assumption achieves* (t,ε)*-security with* $t = \mathsf{poly}(\lambda)$ *and* $\varepsilon = 0$ *in the generic model of multilinear maps.*

Proof. The proof is similar to the analysis in Appendix C. In fact, the extractable security implies the soundness security since the probability that the extractor can extract a witness is 0 when the subset sum does not exist. The main difference is that if the target encoding $g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}$ can be constructed then the subset-sum for $\mathbf{s}$ exists which contradicts with the fact that $x \notin L$. We can easily see that $Pr\left[D\left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, g_{\mathbf{s}}^{\alpha^{\mathbf{s}}}\right) = 1\right] = Pr\left[D\left(\mathsf{param}, \left\{g_{\mathbf{v}_i}^{\alpha^{\mathbf{v}_i}}\right\}_{i\in I}, g_{\mathbf{s}}^{r}\right) = 1\right]$.

E 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 10 (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 $f_1, \cdots, f_k$. We first construct a graph G from f by setting:

$$\begin{aligned} V(G) &= \{\, x_i \mid 1 \leq i \leq n \,\} \cup \{\, \overline{x_i} \mid 1 \leq i \leq n \,\} \cup \{\, f_j \mid 1 \leq j \leq k \,\} \\ E(G) &= \{\, x_i \overline{x_i} \mid 1 \leq i \leq n \,\} \cup \{\, x_i f_j \mid x_i \in f_j \,\} \cup \{\, \overline{x_i} f_j \mid \overline{x_i} \in f_j \,\} \end{aligned}$$

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:

$$\begin{aligned} X &= \{\, f_j \mid 1 \leq j \leq k \,\} \cup E(G) \\ \mathcal{A} &= \{\, E(x_i) \mid 1 \leq i \leq n \,\} \cup \{\, E(\overline{x_i}) \mid 1 \leq i \leq n \,\} \cup \{\, \{f_j\} \cup F_j \mid F_j \subset E(f_j), 1 \leq j \leq k \,\} \end{aligned}$$

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 $2n + 7k$.

F Instantiation and Plausibility

In order to encrypt with witness encryption, we write C program for bitcoin mining procedure, translate the C program into CNF clauses by using the tool CBMC [18]. 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 proof-of-work constraints (as described in Figure 4) for 5 linked block headers. In the "hashPrevBlock" field of the first block header, we put the hash value of the Block 350108 [1]. 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 nondeterministic 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.

The size of the resulting CNF formula is given in Figure 5. We use the tool Coprocessor [33], 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 5a and Figure 5b, 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 later. We also use the Coprocessor [33], a CNF simplifier, to quickly simplify the CNF formula generated from CBMC. Coprocessor has a function that can exclude specified variables from the simplification, which preserves the equality of the formula w.r.t. those variables during the simplification.

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

Fig. 5: Size of CNF clauses for bitcoin mining procedure

Since the breakthrough construction of Garg, Gentry and Halevi [23] in 2013, multilinear maps [23, 19, 31, 20] becomes a very active research area, as well as its cryptanalysis [16, 29, 38, 17]. Our design of witness encryption is independent of the underlying implementations of multilinear maps. In particular, our scheme is not susceptible to the zeroising attacks, as we do not publish low level zero encodings.

We instantiate our witness encryption with CLT13 multilinear maps [19] in order to justify the plausibility of our timed-release encryption. We 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 [16] on CLT13 maps heavily rely on a sufficient amount of low-level encodings of zero and level-0 encodings which are not explicitly published in applications like witness encryption and program obfuscation because these applications do not need the re-randomisation of the encodings. Our design of witness encryption is independent of the underlying implementations 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 CLT13 maps generates n secret primes p_i and publishes $x_0 = \prod_{i=1}^{n} p_i$, 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 $1 \leq i \leq n$:

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

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. Suppose the top-level index is $\mathbf{s} := (s_1, \cdots, s_\ell)$. To enable the zero-testing on the top level, CLT13 scheme publishes a zero-testing parameter

$$P_{zt} = \sum_{i=1}^{n} h_i \cdot \left(z_1^{s_1} \cdots z_\ell^{s_\ell} \cdot g_i^{-1} \bmod p_i \right) \cdot \prod_{j \neq i} p_j \bmod x_0$$

for some small integers h_i. Based on the zero-testing procedure, one can define an extractor $\mathsf{Extract}_s(\mathsf{msbs}_\nu(c \cdot P_{zt} \bmod x_0))$ to extract a random function of the vector $\mathbf{m}$ encoded in the level-$\mathbf{s}$ encoding c. This extraction procedure outputs a canonical representation from the encoding and actually gives the symmetric encryption key.

The decryption involves $n+m-k$ mapping operations as analysed in Section 2.3, hence the multilinearity level is also $n+m-k$. When the instantiations of the multilinear maps are not compact, e.g., [23, 19, 31], that is, the size of elements in the groups depends on the multilinearity κ, we can balance the size of the ring elements and the number of the ring elements in the encoding as below. Instead of encoding $\mathbf{v}_{j,1}$ as $g_{\mathbf{v}_{j,1}}^{\alpha^{\mathbf{v}_{j,1}}}$, we can encode $\mathbf{v}_{j,1}$ as $2\lfloor \mathsf{log}(m_j) \rfloor$ elements:

$$g_{\mathbf{v}_{j,1}}^{\alpha^{\mathbf{v}_{j,1}}}, g_{2\mathbf{v}_{j,1}}^{\alpha^{2\mathbf{v}_{j,1}}}, g_{4\mathbf{v}_{j,1}}^{\alpha^{4\mathbf{v}_{j,1}}}, \cdots, g_{d\mathbf{v}_{j,1}}^{\alpha^{d\mathbf{v}_{j,1}}}$$

where $d = 2^{\lfloor \mathsf{log}(m_j) \rfloor}$. As a result, this will keep the multilinearity as $n + \sum_{j=1}^{k} \mathsf{log}(m_j)$, instead of $n + \sum_{j=1}^{k} (m_j - 1)$, while the size of the encoding is of $2n + 2\sum_{i=1}^{k} \lfloor \mathsf{log}(m_j) \rfloor$ group elements. However, this optimisation introduces some encodings of zeroes. We leave it as a future work to see whether this optimisation is secure in the current instantiations of multilinear maps.

According to the suggestions of parameters settings for CLT13 maps [19], 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. The current multilinear maps are not yet practical. For the application of witness encryption, the time for encryption and decryption is astronomical. However, this time-delay is independent of the time-delay introduced in bitcoin mining procedure. Essentially, the efficiency of the encryption and decryption algorithm is in (big) polynomial time while the Bitcoin mining is in exponential time. The former is due to the state-of-art technology which will be improved in future, while the latter can be tuned by hand. That is, the time for encryption and decryption is not necessarily longer than the time of bitcoin mining. To illustrate this point, assume there is only one block involved in our timed-release encryption. According to Figure 5 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 $a \cdot b (\mathsf{mod}\ x_0)$ has time complexity $O\big(\mathsf{log}^2(x_0)\big)$. 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 [19]. 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 5, which will not affect the decryption time.