

# スマートコントラクトはデジタル署名を生成できるか？

面 和成<sup>1,a)</sup>

**概要：**スマートコントラクトがブロックチェーン上の自動処理を目的として頻繁に使われている。また、スマートコントラクトに関する研究が盛んに行われている。しかしながら、スマートコントラクトは透明性を満たすことから秘密情報を保持できないという制約を持つため、スマートコントラクトは秘密情報を用いた計算ができない。特に、秘密鍵を用いた計算であるデジタル署名の生成が一般にできない。このことがスマートコントラクトの可能性を限定的している。本研究では、スマートコントラクトが秘密鍵を保持していない場合でもスマートコントラクトが署名を生成できる新たな方法を提案する。提案方式はスマートコントラクトが署名鍵を保持しているオフチェーンサーバ（TEEがサポートされていないサーバ）に署名生成をセキュアに委任できる。たとえオフチェーンサーバが攻撃者に乗っ取られて、秘密鍵（署名鍵）が盗まれたとしても、スマートコントラクトが正当な署名を生成する。我々は、スマートコントラクトとインセンティブメカニズムを用いることによって、署名鍵が公開されても安全かつ効率的に署名を生成できる新しいアプローチを提供する。

**キーワード：**ブロックチェーン、スマートコントラクト、署名委任、インセンティブメカニズム

## Are Smart Contracts Capable of Secure Signing?

KAZUMASA OMOTE<sup>1,a)</sup>

**Abstract:** This study proposes a new method that allows smart contracts to generate signatures even without holding secret keys. The proposed method enables smart contracts to securely delegate signature generation to an off-chain server (a server that does not support TEE) that holds the signing key. Even if the off-chain server is compromised by an attacker and the secret key (signing key) is stolen, the smart contract can still generate valid signatures. We provide a new approach that enables secure and efficient signature generation even when the signing key is publicly disclosed, by utilizing smart contracts and incentive mechanisms.

**Keywords:** blockchain, smart contract, delegated signature, incentive mechanism

## 1. はじめに

### 1.1 背景

スマートコントラクトは、ブロックチェーン上で自動的に実行されるプログラム可能な契約であり、信頼性の高い取引や業務の自動化を実現する。その仕組みはデジタル署名と密接に関係しており、取引やスマートコントラクトの操作はユーザの秘密鍵で署名されるため、それらの正当性

が保証される。デジタル署名とスマートコントラクトの連携により、従来の契約手続きの煩雑さを解消し、信頼性と効率性を兼ね備えた新しい取引基盤を提供する。スマートコントラクトは、デジタル経済の発展を支える重要な技術であり、研究も盛んに行われている[15, 16]。

ユーザの依頼を受けてスマートコントラクトが署名を生成する状況を考える。これができるようになると、中央の管理者を必要としない、自律的な取引や認証の仕組みを構築することが可能になる。例えば、契約の締結において、中央の管理者が介在しなくとも、スマートコントラクトが取引内容を確認し、適切な条件が満たされた場合に自動的

<sup>1</sup> 筑波大学  
University of Tsukuba  
a) omote@risk.tsukuba.ac.jp

に署名を行うことで、安全かつ効率的なシステムを実現できる。さらに、スマートコントラクトが自身で生成した署名を保持しておくことで、その署名を他のスマートコントラクトやユーザが利用できるというメリットもある。例えば、ある取引の正当性を証明するために、別のスマートコントラクトがその署名を検証することができる。これにより、複数のスマートコントラクトが連携して動作するような複雑な仕組みも実現しやすくなる。

スマートコントラクトは透明性を重視する設計のため、スマートコントラクトが扱う全てのデータは基本的に公開される。この性質により、スマートコントラクトは秘密情報を保持することが難しく、署名生成など、秘密鍵（署名鍵）を必要とする操作が行えないという課題を持つ。そのため、オフチェーンの計算機が署名生成を代理で実行し、その結果をスマートコントラクトに提供する方法が必要である。スマートコントラクトで署名生成を可能にする技術開発は、さらなる応用範囲の拡大に向けた重要な課題である。

スマートコントラクトが署名生成の計算をオフチェーンの計算機に委任する際、その計算機における秘密鍵の管理は極めて重要になる。秘密鍵が攻撃者に盗まれた場合、偽造署名が作成され、システムの信頼性が大きく損なわれる可能性がある。一方、秘密鍵を物理的に厳重に管理するには、高度なセキュリティ対策が必要であり、それには大きなコストがかかる。この課題を根本的に解決するには、秘密鍵を厳重に保管する必要がない新たな仕組みが効果的である。

スマートコントラクトにおける計算をオフチェーンの計算機に委任する際、インセンティブメカニズムは重要な要素である。オフチェーンの計算機が計算を代行するには、適切な動機づけが不可欠である。これがないとその計算機が非協力的な態度を取るリスクが高まる。インセンティブの提供によって、計算機の管理者は計算処理を行う対価を得られるため、積極的な参加が促される。さらに、インセンティブを導入することで、複数の計算機が競争的に参加する環境が生まれる。この競争原理により、計算機が計算処理を迅速に行おうとする動機が高まり、署名生成やその他の計算が高速化されることが期待される。例えば、分散型オラクルネットワークである Chainlink [1] は、トーカン報酬を提供してノードの積極的な参加を促す仕組みを持っている。

## 1.2 関連研究

スマートコントラクトで秘密鍵を保持できないという制約から、スマートコントラクトが署名生成をオフチェーンの計算機に委任し、その結果を安全に受け取る研究や技術開発が盛んである。このとき、スマートコントラクトは生成された署名を検証する。署名検証は公開鍵のみを用いて

行われるため、署名の自動検証はスマートコントラクトの得意とする機能である。オフチェーンの計算機に安全に署名計算を委任する文脈において、スマートコントラクトが署名を自動検証する仕組みがよく使われている [8, 10, 19]。また、署名計算の委任以外においても、スマートコントラクトを用いて署名検証を自動実行する方法が頻繁に使われている [5, 7, 12, 17, 18, 20]。

しかし、オフチェーンの計算機で署名生成の計算を行う場合、オフチェーンの計算機にある秘密鍵が攻撃者によって盗まれると、一般に偽造署名が生成される恐れがある。また、オフチェーンの計算機が攻撃者に乗っ取られた場合、署名すべきメッセージが不正なものに差し替えられるリスクがある。そこで、これらの問題を解決するために、Trusted Execution Environment (TEE) を用いた方法が研究されている [2, 4, 11]。これは Intel SGX などのハードウェアベースの信頼環境 (TEE) を活用した署名生成のアプローチである。この方法では、秘密鍵を安全な環境内 (TEE 内) に保持し、必要に応じて署名生成を実行する。署名生成のリクエストがオフチェーンサーバに送信されると、サーバは TEE を利用して安全に署名を生成し、その署名をブロックチェーン上に送付する。この仕組みにより、秘密鍵の漏洩リスクを最小化しながら、効率的で信頼性の高い署名生成の委任が可能になる。

Krenn ら [10] は、スマートコントラクトが秘密鍵を保持する代わりに、スマートコントラクトの管理者（または代理人）が署名生成処理を行う方式を提案している。スマートコントラクトの管理者がゼロ知識証明を使って、秘密鍵を公開することなく署名生成処理を実行できる。これにより、スマートコントラクトが秘密鍵を管理する必要がなくなる。また、署名計算の委任に類似するものとして復号計算をオフチェーンの計算機に委任する機能を実現した研究がいくつか存在する [3, 6, 9]。

ただし、これらの方法は全て秘密鍵の安全な管理を要求しており、秘密鍵が盗まれないようにするために、高度な暗号技術やセキュアハードウェアが利用されている。つまり、秘密鍵が漏洩しないような対策が施されている方式であり、そもそも我々のアプローチとは異なる。

なお、Omote [13] は、スマートコントラクトで署名生成を行う際に、署名生成関数を用いることなくスマートコントラクトが署名を生成できる可能性について議論している。この研究では、署名生成関数を用いないため秘密鍵（署名鍵）は不要であり、スマートコントラクトが出力するレシートが署名の代わりになることを主張している。

## 1.3 貢献

本研究<sup>\*1</sup>では、スマートコントラクトが秘密鍵（署名鍵）

<sup>\*1</sup> 本論文は国際会議 IEEE BCCA 2025 [14] で発表予定である。

を保持していないスマートコントラクトが署名を生成できる新たな方式を提案する。提案方式はスマートコントラクトが署名鍵を保持しているオフチェーンサーバ（TEEがサポートされていないサーバ）に署名生成をセキュアに委任できる。たとえオフチェーンサーバが攻撃者に乗っ取られて秘密鍵が盗まれたとしても、スマートコントラクトが正当な署名を生成する。さらに、実装を行うことによって提案方式のガスコストを評価し、本方式が実現可能であることを主張する。

提案方式の主要な貢献は次の通りである。

- 我々は、スマートコントラクトとインセンティブメカニズムを用いることによって、署名鍵が公開されても安全かつ効果的に署名を生成できる新しいアプローチを提供する。オフチェーンサーバは、公開可能な署名鍵を使用して署名を計算するため、鍵を秘密に保持する必要がない。この仕組みにより、サーバが攻撃を受けたとしても、鍵の漏洩がセキュリティに影響を与えることはなく、偽造署名が生成される心配がない。
- 署名生成スマートコントラクトは、自身で署名鍵を持たずとも、オフチェーンサーバに署名生成をセキュアに委任することで署名を生成できる。この仕組みにより、スマートコントラクトが秘密情報を直接管理する必要がなく、セキュリティリスクを軽減しながら署名生成の機能を実現する。
- 複数のオフチェーンサーバを利用し、インセンティブを与える仕組みを導入することで、サーバ同士が競争する環境が作られる。この競争により、各サーバがより早く署名計算を行おうとするため、署名生成にかかる遅延を最小限に抑えることができる。これにより、システム全体の効率性が向上し、迅速な署名生成が可能になる。

## 2. スマートコントラクト

スマートコントラクトは、ブロックチェーン上で実行される自己実行型のプログラムであり、事前に定義された条件が満たされると自動的に実行される。この特性により、第三者を介さずに処理を行えるため、仲介コストを削減し、手続き時間を短縮する利点がある。また、耐改ざん性、透明性、可用性を備えているため、金融取引、サプライチェーン管理、デジタルアイデンティティ管理など幅広い分野での応用が進んでいる。特に、スマートコントラクトはデジタル署名と密接に関係しており、署名検証の基盤として重要な役割を果たす。たとえば、Ethereumでは、プリコンパイルコントラクトである署名検証関数（例：ecrecover）を使用して、取引発信者の正当性を確認できる。このような仕組みにより、権限管理や不正取引防止が効率化され、認証プロセスを簡略化できる。これらの特徴により、スマート

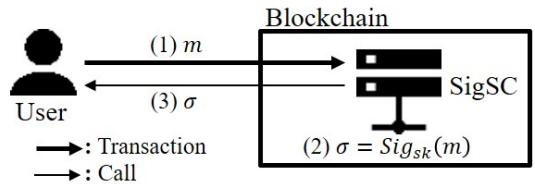


図 1: スマートコントラクトを用いて署名を生成する概略図

トコントラクトは、署名ベースのシステムにおける信頼性の高いツールとして広く活用されている。

一方で、署名生成について見てみると、図1はスマートコントラクトを用いた署名生成の概念図を示している。ユーザは、メッセージ  $m$  のトランザクションを発行して、署名生成スマートコントラクト (SigSC) に送付する。SigSC は署名鍵  $sk$  を用いて署名  $\sigma$  を計算する。ユーザは Call 处理でブロックチェーンから  $m$  に対する署名  $\sigma$  を取得する。しかしながら、スマートコントラクトは秘密情報を保持する設計にならないため、秘密鍵を使用した直接的な署名生成は不可能である。その代わりに、1.2節で述べたように、ブロックチェーン外の信頼できるシステムやプロトコルを活用する代替アプローチが数多く研究されている。さらに、スマートコントラクトにおける署名生成はプリコンパイルコントラクトとして実装されていないため、例えば ECDSA 署名生成におけるガスコストは署名検証に比べて非常に高くなることが予想される。このため、署名生成をブロックチェーン外部の計算機に委任するアプローチは合理的といえる。

## 3. 提案方式

本章では、スマートコントラクトが秘密鍵を保持せずともスマートコントラクトが署名を生成できる新たな方式について詳しく説明する。署名鍵を保持しているオフチェーンサーバに対してスマートコントラクトが署名生成をセキュアに委任する。図2は、我々のスマートコントラクトによる委任署名システムの全体図である。

### 3.1 解決すべきチャレンジ

スマートコントラクトが署名を生成する際、秘密鍵をオフチェーンに置けないため、一般に署名生成の計算をオフチェーンの計算機に委任する方法が研究されている。しかし、オフチェーンの計算機への委任にはいくつかの課題がある。第一に、秘密鍵（署名鍵）の管理が最大の問題となる。秘密鍵が攻撃者に盗まれると、一般に偽造署名が生成され、不正な取引が承認される危険性がある。この問題を解決するためには、署名鍵を秘密に保持する必要がない革新的なフレームワークが重要である。第二に、オフチェーン計算では、通信遅延や計算待ち時間によって、スマートコントラクトの全体的な処理速度が低下する懸念がある。

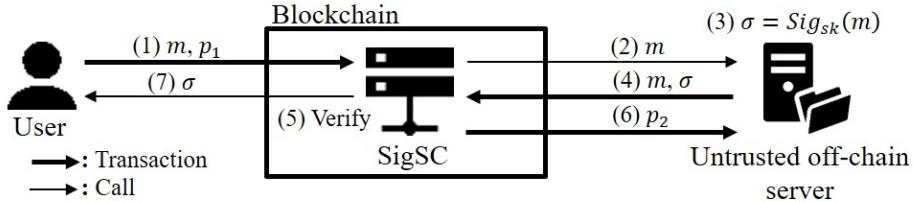


図 2: スマートコントラクトによる提案代理署名システムの全体図

この遅延を最小限に抑えるため、オフチェーンの計算機にインセンティブを与えるメカニズムの設計が不可欠である。例えば、オフチェーンサーバに計算結果に応じた報酬を提供することで、サーバ間の競争を促し、迅速かつ正確な計算を実現できる。

### 3.2 コアアイデア

我々は、スマートコントラクトとインセンティブメカニズムを用いることによって、署名鍵が公開されていても安全に署名を生成できる新しいアプローチを提供する。提案方式では、署名生成スマートコントラクト (SigSC) が秘密鍵を持たず、署名生成をセキュアに委任できる仕組みを提案する。この方式の最大の特長は、オフチェーンサーバにある署名鍵が公開されていても問題がない点にある。オフチェーンサーバは、秘匿する必要がない署名鍵を使って署名計算を行う。そのため、たとえオフチェーンサーバが攻撃を受けて秘密鍵が盗まれたとしても、偽造署名が生成されるリスクが最小限になる。これにより、オフチェーンサーバのセキュリティ要件が緩和されると同時に、安全性と信頼性を高めた署名生成プロセスを実現する。なお、SigSC が出力した署名のみ正当な署名となることに注意する。

SigSC はサーバから正当な署名を受け取ると、あらかじめ定義されたルールに従い、複数のオフチェーンサーバの中での勝者（正当な署名が最初にアクセプトされたノード）に対して自動的にインセンティブ（暗号資産）を支払う。このインセンティブメカニズムは、計算遅延や通信待ち時間を最小限に抑えるための重要な役割を果たす。インセンティブがあることで、オフチェーンサーバ間での競争が促進され、迅速かつ正確な計算が期待される。

このように、提案方式は、セキュリティ、効率性、そして分散型システムの特性を活用したインセンティブメカニズムを組み合わせることで、従来の方法では解決が難しかった署名生成の課題に対応している。これにより、スマートコントラクトとオフチェーンサーバの連携が最適化され、より多くのユースケースに対応できる可能性が広がる。

### 3.3 表記

提案方式で用いる表記は表 1 の通りである。

表 1: 表記

記号	説明
$m$	メッセージ
$\sigma$	署名
SigSC	署名生成スマートコントラクト
$(sk, pk)$	秘密鍵と公開鍵のペア
$\text{Sig}_{sk}(m)$	$m$ に対して $sk$ で署名を生成する関数
$\text{Ver}_{pk}(m, \sigma)$	$m$ に対して $pk$ で署名 $\sigma$ を検証する関数
$T_1$	$m$ を格納するブロックチェーン上のテーブル
$T_2$	$(m, \sigma)$ を格納するブロックチェーン上のテーブル
$p_1$	署名生成費用として SigSC に送付する暗号資産
$p_2$	署名生成のインセンティブ（暗号資産）
$P$	署名生成に必要な費用（暗号資産）
$addr_U$	ユーザのウォレットアドレス
$addr_{SC}$	SigSC のウォレットアドレス
$addr_S$	オフチェーンサーバのウォレットアドレス

### 3.4 エンティティ

我々は、本節で本システムに登場する 3 つのエンティティについて説明する。

- SigSC は、署名生成スマートコントラクトであるが、実際は署名生成の計算をオフチェーンサーバに委任する。このとき、ノードからのトランザクションを受けとて処理を行う。内部でメッセージ  $m$  を保存するテーブル  $T_1$ 、及びメッセージと署名のペアを保存するテーブル  $T_2$  を持つ。また、スマートコントラクトはウォレットとそのアドレス  $addr_{SC}$  を持ち、インセンティブとして暗号資産をオフチェーンサーバに自動送信する。
- ユーザはウォレットアドレス  $addr_U$  を持つ。SigSC に対して、メッセージ及び暗号資産のトランザクションを発行したり、コールして  $m$  に対応する署名  $\sigma$  を取得したりする。
- オフチェーンサーバは TEE をサポートしていない信頼できないサーバであり、ウォレットアドレス  $addr_S$  を持つ。SigSC から  $m$  を取得してその署名  $\sigma$  を生成する。このとき、秘密鍵  $sk$  を用いて署名するが、 $sk$  は秘匿しておく必要がない。それから、サーバは  $(m, \sigma)$  をトランザクション経由で SigSC に送付し、受理されると署名計算のインセンティブを受け取る。サーバは SigSC によって認証される必要はなく、任意のサーバがオフチェーンサーバとして参加できる。また、サー

バと SigSC の間はセキュチャネルが不要である。

### 3.5 攻撃モデル

ユーザ、ブロックチェーン、スマートコントラクトが信頼できると仮定して、次の 2 つの攻撃モデルを考える。

- 署名の偽造攻撃：攻撃者は、秘密鍵を不正に取得して、ユーザがこれまで SigSC にリクエストしていないメッセージに対する偽造署名を作成しようとする。
- なりすまし攻撃：ユーザとオフチェーンサーバの 2 種類のなりすましを考える。ユーザのなりすまし攻撃では、攻撃者が正規のユーザになりすまし、スマートコントラクトに対して不正に署名生成を依頼しようとする。また、オフチェーンサーバのなりすまし攻撃では、攻撃者が秘密鍵を不正に取得し、正規のサーバに代わって署名を生成しようとする。

### 3.6 手順

図 2, 3 に基づいて、スマートコントラクトによる代理署名システムの手順について詳しく説明する。

- (1) ユーザはトランザクションを用いて、 $m, p_1$  を SigSC に送付する。SigSC は、 $p_1 \geq P$  の場合に限り  $m, p_1$  を受理し、その後、 $m$  を  $T_1$  に格納し、 $p_1$  を SigSC のウォレットに保管する。
- (2) サーバは SigSC に対して Call して最も古い  $m$  を  $T_1$  から一つ取得する。
- (3) サーバは  $sk$  を用いて署名  $\sigma = \text{Sig}_{sk}(m)$  を計算する。
- (4) サーバはトランザクションを用いて  $(m, \sigma)$  を SigSC に送付する。
- (5) SigSC は  $m$  が  $T_1$  に含まれていること、 $\text{Ver}_{pk}(m, \sigma) = 1$  になることを検証する。検証が成功したら、 $(m, \sigma)$  を  $T_2$  に格納し、 $T_1$  からその  $m$  を削除する。
- (6) SigSC はインセンティブとして  $p_2$  を自動的にサーバに送金する。一般的に、 $p_2 \leq P$  を満たす。
- (7) ユーザは  $m$  をキーに SigSC に対してコールし、対応する  $\sigma$  を  $T_2$  から取得する。

### 3.7 インセンティブメカニズムを用いた委任計算

SigSC は正当な署名を受け取ると、あらかじめ定義されたルールに従い、オフチェーンサーバの中の勝者に対して自動的にインセンティブ（暗号資産）を支払う。このインセンティブメカニズムは、計算遅延や通信待ち時間を最小限に抑えるための重要な役割を果たす。

具体的には、SigSC が  $m$  をブロックチェーン上に公開すると、それを監視している複数のオフチェーンサーバが  $m$  をダウンロードする。それから、 $m$  に対して署名鍵  $sk$ （署名鍵はオフチェーンサーバ間で共通）で署名を行う。その後、トランザクションを用いて  $(m, \sigma)$  を SigSC に送付する。 $(m, \sigma)$  が SigSC に最も早く受理されたオフチェー

ンサーバが勝者となる。SigSC は勝者のウォレットアドレスが分かるので、そこに対してインセンティブの暗号資産  $p_2$  を自動的に送付する。もし 2 番目に  $(m, \sigma)$  が SigSC に送られてきた場合、たとえ正当な署名であっても SigSC は  $(m, \sigma)$  をリジェクトする。なお、オフチェーンサーバの参加、離脱は自由に行える。

## 4. 実証実験

提案方式が確実に機能するためには、スマートコントラクト SigSC の実装と動作確認が不可欠である。特にトランザクションの発行、2 種類のテーブル操作、Call 操作、インセンティブの自動送付など、主要な機能が正しく動作することを確認する必要がある。また、最初に送付された  $(m, \sigma)$  のみを SigSC が受理する仕様が正しく機能することも確認する。これらの実装を通じて、提案方式の信頼性や実用性を検証する。

### 4.1 環境

我々は Ethereum のテストネットである Sepolia を採用し、solidity v0.8.0 を用いて SigSC を実装する。トランザクションを介したユーザからの  $m$  の送付、トランザクションを介したサーバからの  $(m, \sigma)$  の送付、Call 処理は Python3.9 を用いてそれぞれ実装する。ユーザやサーバからのトランザクション及び Call 処理は Infura サービス（Infura の API など）を使ってブロックチェーンに送付される。SigSC の Solidity コードの行数は約 100 行であり、コード内には 2 種類のテーブルとして追加可能な配列を用いる。ユーザが  $m, p_1$  を送付する Python コードの行数は約 40 行、サーバが Call 処理を行った後、 $(m, \sigma)$  を送付する Python コードは約 50 行、最後にユーザがコール処理する Python コードの行数は約 30 行である。ハッシュは SHA3 を使用し、署名は簡便性を重視し、ECDSA（Elliptic Curve Digital Signature Algorithm）を採用する。トランザクションの処理内容は Ethereum エクスプローラである Etherscan を用いて誰でも確認できる。

### 4.2 実験の流れ

SigSC の管理者は Ethereum テストネットに接続し、SigSC をデプロイする。ユーザとサーバのための 2 種類の外部所有アカウント（EOA）及び SigSC を用いて、図 3 で示される動作フローを再現しながら実験を行い、SigSC に要するガスコストを計測する。ユーザはトランザクションを介して  $m$  を 3 回送付し、 $T_1$  に 3 つの  $m$  を保存する。 $m$  を 3 回送付するのは、複数の  $m$  を保存できることと、 $m$  の保存順序を記録できることを確認するためである。引き続き、サーバはコール処理によって最も古い  $m$  を  $T_1$  から取得し、その署名  $\sigma$  を生成できること、 $(m, \sigma)$  を SinSC に送付できることを確認する。SigSC が最初に送付された

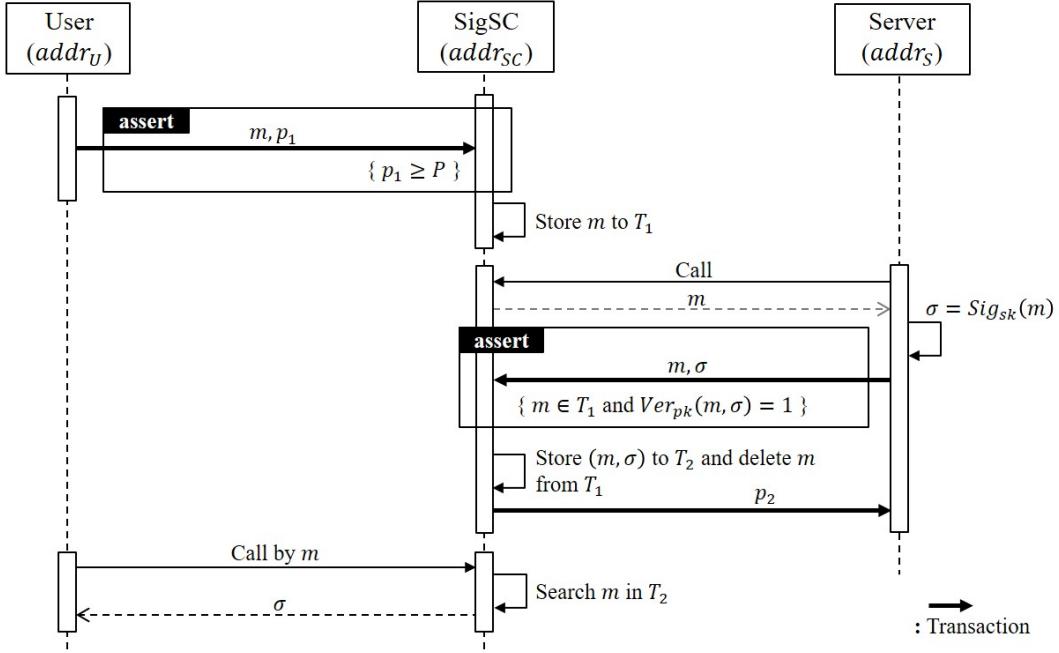


図 3: 提案代理署名システムのシーケンス図

$(m, \sigma)$  のみを受理することを確認するために、別のサーバからの 2 回目のトランザクションが拒否されることを確認する。また、SigSC が署名検証に成功したらインセンティブ  $p_2 = 0.0005\text{ETH}$  をサーバに自動的に送付できることを確認する。それぞれのガスコストは Etherscan に記録され、誰もが確認できる。

#### 4.3 結果

SigSC のデプロイに要したガスコストは  $0.043\text{ETH}$  であった。ユーザが発行するトランザクションに要したガスコストは  $0.0025\text{ETH}$ 、サーバが発行するトランザクション及びインセンティブ送付のインターナルトランザクションに要した総ガスコストは  $0.0082\text{ETH}$  であった。一般に ECDSA の署名生成に必要なガスコストが数百万ガスと見積もられていることから、SigSC のトータルガスコストはリーズナブルである。ただし、Ethereum が高騰しているため、ガスコストが非常に高くなっていることに注意が必要である。

図 4 は、インセンティブの自動送信が実行されていることを Etherscan 上で示している。トランザクションがサーバ (address: 0x37E7...) から SigSC (address: 0xDE70...) に送付され、さらにインセンティブがインターナルトランザクションを介して SigSC からサーバに  $0.0005\text{ETH}$  だけ送付されている。ちなみに、Gas Price は  $50\text{Gwei}$  に設定している。

[ This is a Sepolia Testnet transaction only ]	
① Transaction Hash:	0x1ba4136809a3af1548876f41cc47678c23901092bb7cacf6389587f62593a814
② Status:	<span style="color: green;">Success</span>
③ Block:	7446455 23 Block Confirmations
④ Timestamp:	5 mins ago (Jan-08-2025 11:56:36 AM UTC)
⑤ Transaction Action:	Call 0x3832a0ad Method by 0x37E7DFFC...a28153fD6 on 0xDE70Fd85...80A0Ef256
⑥ From:	0x37E7DFFC9a5aEEe5d0A4f161f09110a28153fD6
⑦ To:	0xDE70Fd8528DAcFE73dF5c960Da427CE80A0Ef256 Transfer 0.0005 ETH From 0xDE70Fd85...80A0Ef256 To 0x37E7DFFC...a28153fD6
⑧ Value:	0 ETH
⑨ Transaction Fee:	0.00818655 ETH
⑩ Gas Price:	50 Gwei (0.00000005 ETH)
More Details:	+ Click to show more

図 4: Ethereum テストネット上で自動化されたインセンティブ送金の取引履歴 (Explorer で表示)

## 5. 考察

### 5.1 セキュリティ

署名の偽造に対する安全性について議論する。提案方式では、仮に攻撃者が秘密鍵  $sk$  を不正に取得したとしても、スマートコントラクト SigSC は偽造署名を出力しない。具体的には、攻撃者が盗んだ秘密鍵  $sk$  を使用して任意の  $(m', \sigma')$  を生成する場合を考える。ただし、正規のユーザによって  $m'$  に対する署名がリクエストされていないものとする。このとき、 $(m', \sigma')$  が SigSC に送信され、検証関数  $Ver(m', \sigma') = 1$  が成立する。しかし、 $m'$  がテーブル T1 に存在しないため、SigSC はこの署名を拒否する。提案方式では、SigSC が送出した署名のみを正当な署名として扱う仕組みを持つ。このアプローチにより、秘密鍵が漏洩

した場合でもシステムの安全性が保たれることを保証する。

次に、なりすまし攻撃に対する安全性について議論する。ユーザになりますためには、ユーザに代わって  $(m, p_1)$  のトランザクションを発行する必要がある。しかし、このトランザクションを発行するためには、ユーザアドレス  $addr_U$  に対応する秘密鍵が必須である。そのため、ユーザの秘密鍵が漏洩しない限り、攻撃者がユーザになりますことは非常に困難である。一方、サーバのなりすましについては、脅威として大きな問題にならない。ここで攻撃者がサーバを乗っ取って秘密鍵  $sk$  を入手した場合を考える。ユーザから依頼があった  $m$  に対する署名であれば正当な署名とみなされ、それ以外の  $m$  に対する署名は SigSC に受理されない無効な署名になるだけである。したがって、提案方式はユーザおよびサーバに対するなりすまし攻撃に対して安全であると結論づけられる。

## 5.2 効率性

SigSC のガスコストの削減は重要な課題である。一般に、スマートコントラクトは署名検証機能のプリコンパイルコントラクトを持つが、署名生成機能のプリコンパイルコントラクトを持たない。そのため、署名生成のガスコストが非常に高くなる。提案方式では、署名生成の計算をオフチェーンサーバに委任する構造であり、SigSC は署名の検証のみを担当する。この設計により、全体のガスコストを効果的に抑えることが可能である。したがって、署名生成をアウトソースする意義は、スマートコントラクトによる署名生成の実現とガスコストの削減の 2 つが挙げられる。

Ethereum ではガスコストが高いことに注意しなければならない。有名なブロックチェーンを使うと、ガストコストが相当高くなるため、ガスコストの安い Polygon (Ethereum のサイドチェーン) など他のブロックチェーンを用いることを検討する必要がある。もし Polygon を使えば、ガスコストは Ethereum の数百分の 1 で済む (2025 年 7 月時点)。

## 5.3 ユースケース

提案方式は、スマートコントラクトにおける署名の自動生成に関する技術であるため、メッセージ  $m$  をブロックチェーン内で保存したり利用したりしつつ署名が必要なユースケースが有用である。その中でも代表的な例として挙げられるのが、文書の自動認証である。あるスマートコントラクトが、別のスマートコントラクトに保存されている署名データを取得し、その署名が正当なものであるかを検証することで、メッセージ認証を自動的に実行できる。例えば、文書のデジタル署名をスマートコントラクトに保存し、取引の際に別のコントラクトがその署名の正当性を確認することで、契約の履行状況を検証できる仕組みが実現できる。

## 5.4 制約

提案方式では、ブロックチェーン上のオンライン環境で署名検証を行うことを想定する。もしオフライン環境で利用する場合、スマートコントラクトにアクセスして署名検証する必要がある。なぜなら、SigSC から出力された署名のみが有効とみなされるためである。さらに、この方式は広く使われているブロックチェーンプラットフォームである Ethereum 上で実装されているため、計算とトランザクションのオーバーヘッドによるガスコストの上昇は避けられない。

## 6. おわりに

本研究では、スマートコントラクトが秘密鍵（署名鍵）を保持していないスマートコントラクトが署名を生成できる新たな方式を提案した。提案方式では、スマートコントラクトが署名鍵を保持しているオフチェーンサーバ (TEE がサポートされていないサーバ) に署名生成をセキュアに委任できる。また、Ethereum テストネット上で実装評価を行うことによってガスコストを計測した。Ethereum ではガスコストが高いため、ガスコストの安い Polygon など他のブロックチェーンを用いることを検討する必要がある。また、通常の署名でも同様であるが、署名検証では  $m$  の内容の正当性をチェックしない。ただし提案方式では、オフチェーンサーバで署名を生成することから、ここで  $m$  の内容をチェックできる可能性がある。これは今後の課題とする。

**謝辞** 本研究は JSPS 科研費 JP23K24844, JP25H01106 の助成を受けたものである。

## 参考文献

- [1] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, 1:1–136, 2021.
- [2] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 185–200. IEEE, 2019.
- [3] H. Cui, Z. Wan, X. Wei, S. Nepal, and X. Yi. Pay as you decrypt: decryption outsourcing for functional encryption using blockchain. *Transactions on Information Forensics and Security*, 15:3227–3238, 2020.
- [4] T. Frassetto, P. Jauernig, D. Koisser, D. Kretzler, B. Schlosser, S. Faust, and A.-R. Sadeghi. POSE: Practical off-chain smart contract execution. In *Network and Distributed System Security Symposium (NDSS)*, pages 1–18, 2023.
- [5] T. Fujitani, K. Emura, and K. Omote. A privacy-preserving enforced bill collection system using smart contracts. In *16th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 51–60. IEEE, 2021.

- [6] C. Ge, Z. Liu, W. Susilo, L. Fang, and H. Wang. Attribute-based encryption with reliable outsourced decryption in cloud computing using smart contract. *Transactions on Dependable and Secure Computing*, 21(2):937–948, 2024.
- [7] S. Haga and K. Omote. Blockchain-based autonomous notarization system using national eid card. *IEEE Access*, 10:87477–87489, 2022.
- [8] J. Hao, C. Huang, W. Tang, Y. Zhang, and S. Yuan. Smart contract-based access control through off-chain signature and on-chain evaluation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(4):2221–2225, 2021.
- [9] Z. Hou, J. Ning, X. Huang, S. Xu, and L. Y. Zhang. Blockchain-based efficient verifiable outsourced attribute-based encryption in cloud. *Computer Standards & Interfaces*, 90:1–12, 2024.
- [10] S. Krenn and T. Lorünser. Single-use delegatable signatures based on smart contracts. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–7. ACM, 2021.
- [11] R. Li, Q. Wang, Q. Wang, D. Galindo, and M. Ryan. Sok: Tee-assisted confidential smart contract. In *Privacy Enhancing Technologies (PoPETs)*, pages 711–721, 2022.
- [12] S. Matsumoto and R. M. Reischuk. IKP: Turning a PKI around with decentralized automated incentives. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 410–426. IEEE, 2017.
- [13] K. Omote. Towards decentralized autonomous digital signatures using smart contracts. In *International Conference on Information Networking (ICOIN)*, pages 230–235. IEEE, 2023.
- [14] K. Omote. Can smart contracts generate digital signatures? In *Proceedings of the 7th International Conference on Blockchain Computing and Applications (BCCA)*, pages 1–7. IEEE, 2025.
- [15] K. Omote, T. Ikeda, E. Kono, N. Mimura, H. Nagai, and N. Kashiwabara. Analysis of papers and patents in the blockchain field using e-csti. In *Proceedings of the 6th International Conference on Blockchain Computing and Applications (BCCA)*, pages 1–8. IEEE, 2024.
- [16] K. Omote, Y. Inoue, Y. Terada, N. Shichijo, and T. Shirai. A scientometrics analysis of cybersecurity using e-csti. *IEEE Access*, 12:40350–40367, 2024.
- [17] C. Patsonakis, K. Samari, A. Kiayias, and M. Rousopoulos. On the practicality of a smart contract pki. In *IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCon)*, pages 109–118. IEEE, 2019.
- [18] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo. Blockchain-based fair payment smart contract for public cloud storage auditing. *Information Sciences*, 519:348–362, 2020.
- [19] W. Xiong and Y. Hu. Delegate contract signing mechanism based on smart contract. *Plos one*, 17(8):1–24, 2022.
- [20] X. Yan, X. Yuan, Q. Ye, and Y. Tang. Blockchain-based searchable encryption scheme with fair payment. *IEEE Access*, 8:109687–109706, 2020.