École Polytechnique Fédérale de Lausanne

Optimizing Front-running Protection

by Julie Bettens

Master Project Report

Approved by the Examining Committee:

Prof. Dr. Bryan Ford
Thesis Advisor

Haoqian Zhang
Thesis Supervisor

EPFL IC IINFCOM DEDIS
BC 210 (Bâtiment BC)
Station 14
CH-1015 Lausanne

June 28th, 2023

Abstract

The F3B architecture [12] provides frontrunning protection for blockchains using threshold cryptography. This work extends F3B with a variant based on an identity-based encryption scheme.

Contents

Chapter 1

Introduction

Since the launch of Bitcoin [8] in 2009, distributed ledger technologies have been an active area of research in the field of decentralized systems and cryptography. Allowing any computer user to safely trade and take ownership of digital assets without relying on an intermediary such as a broker or a bank proved to be an endless source of stimulating research questions. Today, public blockchains, which are distributed ledgers that no single entity has control over, are increasingly used for a variety of purposes. For instance in May 2023, decentralized exchange Uniswap recently self-reported a record of 400'000 monthly active users. [1]

A necessary attribute of public blockchains is that users cannot be prevented from adopting trading strategies that are outlawed in regulated markets. One such strategy is frontrunning, which involves learning about a future trade and executing a trade ahead of it to benefit from the price impact. Not only is this practice harmful to less sophisticated market participants, it has also been argued to pose a risk to the operation of the blockchain itself. [5]

F3B: the solution proposed by Zhang et al., involves users being able to share their transactions in encrypted form, and in such a way that the plaintext will be revealed after the transaction is committed on chain. The timely decryption of the transaction is enforced by the Secret Management Committee (SMC): a properly incentivized group of nodes cooperating using threshold cryptography. [12]

In this work, we implement and evaluate a different cryptographic scheme than the one chosen by Zhang et al. to better understand the possible tradeoffs.

Chapter 2

Background

2.1 Decentralized systems

In most areas of computer science, programmers are allowed to assume that the computer systems their programs will run on will always execute their instructions faithfully. In the area of decentralized computing, this is not the case: programs — *protocols* rather — are expected to be able to operate normally even if some fraction of the computers — known as *nodes* — go offline, or even actively try to sabotage the system. It is more difficult to design systems in this way because the assumption that no nodes are compromised is handy, but on the other hand, decentralized protocols are more resilient to attacks and can operate on environments that other protocols could not.

One such environment is the problem of distributed ledgers or blockchains. In such systems, participants want to be able to perform arbitrarily complex financial transactions without relying on trusted intermediaries to resolve disputes. There are non-financial uses as well. Part of the solution to this challenge is the use of a consensus mechanism, which is a system through which nodes can discover which transactions were processed in which order, without the possibility of any participant removing or reordering transactions long after the fact. This eventually results an immutable, append-only database.

A consensus mechanism typically involves a group of nodes called the Consensus Group (CG), executing a protocol which results in picking a new batch of transactions called a block to add to the append-only ledger. At each round, a new block is selected, and a sequence of consecutive canonical blocks eventually emerges. There is an underlying assumption that no one party controls a threshold of nodes in the CG, otherwise they would be able to arbitrarily manipulate the ledger. As a result, blockchains try to attract a diverse set of nodes to the CG and to incentivize them in some way to act in a way that benefits the protocol.

Due to the decentralized design, users who wish to make transactions have typically been forced to broadcast their transactions to all the nodes in the CG, whom they do not trust. As a result, CG nodes are in a privileged position in the protocol where they can see future transactions and decide with full discretion whether to censor some of them, reorder, or insert their own transactions in such a way that is beneficial to them. In particular, frontrunning can occur.

To mitigate this attack vector, F3B introduces a second group of nodes called the SMC. Transactions are encrypted in such a way that the SMC can decrypt them, and the SMC is tasked with decrypting them after the CG has committed them in a given order. This stops the CG from censoring and frontrunning, unless they are able to collude with the SMC. Mechanisms to reward and punish SMC nodes as needed are also being studied. [12]

2.2 Cryptography

We will now cover in a high-level way the advanced cryptographic building blocks we use in this document.

A Distributed Key Generation (DKG) is a process by which multiple nodes generate a cryptographic key in such a way that no node has access to the full key. Instead, they all keep overlapping shares of the key. As a result, a number t called the *threshold* of nodes need to cooperate to use the key. The key can be for instance a signing key or a decryption key, provided there exists a *threshold cryptosystem* to enable this usage of the key. If there are n nodes, the network can still operate even if $t-1$ nodes are malicious, or $n-t$ nodes are unresponsive, making these protocols very useful in a decentralized setting. [7]

An Identity-based Encryption (IBE) scheme is a system where there exists a trusted *key generation center* which is able to issue decryption keys matching arbitrary labels. The key generation center also broadcasts a public key that allows anyone to derive encryption keys for arbitrary labels. The main idea is that Alice can send an encrypted message to Bob simply by knowing his unambiguous name and deriving the encryption key for it, and Bob can at any point authenticate to the key generation center to receive his decryption key. [9]

2.3 Our contribution

In its first iteration, F3B relies on an implementation of the TDH2 cryptosystem introduced by Shoup and Gennaro. It allows creating ciphertexts that can be decrypted by any t out of n members of the SMC. [10] In this model, the user who wishes to perform a transaction first retrieves the current state of the SMC. Then they symmetrically encrypt the transaction using a random secret

k, and encrypt k so that a threshold of the SMC can decrypt it. The two resulting ciphertexts are then published on-chain. Once the transaction is committed, the SMC cooperatively decrypt k and submit it to the CG, who are thus able to execute the transaction. [12]

Zhang et al. want to explore alternatives to the TDH2-based design. In particular, they wish to implement and evaluate a protocol that uses IBE and derives a decryption identity for each future block based on its height. The key generation center role is performed by the SMC. Importantly, this is less flexible than TDH2 since it does not allow the SMC to enact per-transaction decisions, but it is expected that it has lower communication and computation costs, in particular due to economies of scale.

Chapter 3

Design

The existing F3B implementation is composed of a modified go-ethereum client which plays the role of the CG, and a version of Dela which includes code for TDH2. Go-ethereum communicates with Dela by invoking a program called `dkgcli`. [11] We aim to keep compatibility with this architecture.

Dela already implements a DKG procedure based on "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems"[7] which is used for TDH2. However, it supports the Ed25519 curve group [4] which does not support elliptic curve pairings. Thankfully, the procedure can be adapted to work for other curve groups such as BN256 [3] or BLS12-381 [2] which are suitable for BLS signatures.

Recently, Gailly, Melissaris, and Romailler presented tlock: a practical deployment of time-lock encryption based on IBE and threshold networks. Given a random beacon that produces BLS signatures at a predictable pace, tlock can produce a ciphertext that will only be decryptable at a predetermined time in the future. [6] This scheme can be adapted to our task by having the SMC act as a per-block random beacon by releasing per-block decryption keys once the corresponding block becomes canonical.

We will now describe in a semi-formal way the high-level cryptographic protocol used in this work. It is largely derived from Gailly, Melissaris, and Romailler. [6]

3.1 Full protocol

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups where the computational Diffie-Hellman assumption holds. q their order, g_1, g_2, g_T their respective generators, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a non-degenerate bilinear pairing such that

$e(g_1, g_2) = g_T$, and $\mathrm{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ and $\mathrm{H}_2 : \mathbb{G}_T \to \{0,1\}^l$ cryptographic hash functions. We require that the bilinear Diffie-Hellman problem on e be hard.

Step 0: DKG Setup Before declaring itself ready to manage encrypted transactions, the SMC runs the synchronous DKG protocol proposed by Gennaro et al. [7] to generate a shared public key $\mathrm{cpk} = \mathrm{csk} \cdot g_2$, and shares of the private key for each trustee are denoted as sk_i. The corresponding private key csk can be reconstructed only by combining t private key shares. Specifically, for all $S \subset \{1, \ldots, n\}$ such that $\#S = t$, $\mathrm{csk} = \sum_{i \in S} sk_i \lambda_i$ where λ_i is the ith Lagrange element. We assume that cpk and (i, pk_i) where $\mathrm{pk}_i = \mathrm{sk}_i \cdot g_2$ are written into the blockchain as metadata.

Step 1: Encrypt Transaction Given the plaintext transaction data $\mathrm{tx} \in \{0,1\}^*$, The sender starts the protocol by preparing the ciphertext like so:

1. Derive the encryption key $P = e(\mathrm{H}_1(L||n), \mathrm{cpk})$ from the label of the underlying blockchain L and the target block number n.
2. Pick a random number $r \in \mathbb{Z}_q^*$.
3. Compute the symmetric secret key $k = \mathrm{H}_2(rP)$.
4. Form the ciphertext $(U, V) = (rg_2, \mathrm{enc}_k(\mathrm{tx}))$.

Step 2: Shares Preparation by Trustees Once block n is finalized, each trustee i performs the following steps:

1. Compute $\sigma_i = \mathrm{sk}_i \cdot \mathrm{H}_1(L||n)$.
2. Send (i, σ_i) to the CG.

Step 3: Key Reconstruction Upon receiving t shares, each node in the CG executes the following:

1. For each share, check that $e(\sigma_i, g_2) = e(\mathrm{H}_1(L||n), \mathrm{pk}_i)$
2. Combine shares using interpolation to find $\sigma = \sum \sigma_i \lambda_i$.
3. Verify that σ is correct by checking that $e(\sigma, g_2) = e(\mathrm{H}_1(L||n), \mathrm{cpk})$.

Step 4: Decryption and Execution For each encrypted transaction (U, V), CG nodes continue with the following:

1. Compute $k = \mathrm{H}_2(e(\sigma, U))$ and $\mathrm{tx} = \mathrm{dec}_k(V)$.
2. Interpret tx according to the rules of the blockchain.

Chapter 4

Implementation

The delivered implementation of F3B with IBE is available at `https://github.com/dedis/student_23_dela_f3b_ibe`. It consists of a modified version of Dela, and exposes an executable called `dkgcli` which is used both to set up and perform DKG ceremonies, and to derive decryption keys. As a convenience, it is also able to encrypt arbitrary data and decrypt the resulting ciphertexts.

It is worth noting that the implementation does not enforce the security policies it normally ought to in the real world. This was not a goal of this work. Specifically, it has no notion of time or current block height: it will simply accept any IBE label passed as a command line argument.

The system uses BN256 as an elliptic curve pairing. This is because this configuration was readily available in Dela. The pairing should be able to be changed if desired, e.g. when Kyber provides support for BLS-12-381.[1] To implement H_1 which hashes to a $\mathbb{G}_1$ point, we use the procedure provided by Kyber, which hashes to the x coordinate using SHA256. To implement H_2, we use HKDF-SHA512. To implement enc_k, we use AES256-CTR with an all-zero initialization vector. This ensures that the ciphertext is the same size as the plaintext, and is secure as long as the symmetric key is different for each message, which is by design true except with negligible probability.

Ideally, the implementation should be refactored to use Dela as a dependency instead of using a fork. This is pending on the next release of Dela with Kyber 3.1.0 and the addition of a required method.[2]

A simple demo script is also provided. It creates a 16-member DKG on the local machine, shows the common public key and decryption keys for some fictional blocks, encrypts a transaction with IBE, and decrypts it with the SMC.

[1] `https://github.com/dedis/kyber/pull/487`
[2] `https://github.com/dedis/dela/pull/253`

Chapter 5

Evaluation

To ensure correctness of the software implementation, unit tests and integration tests written for the existing TDH2 have been adapted for the implementation of the IBE variant, and new ad-hoc tests have been written. The code coverage from tests is 96.7% of statements.

To evaluate the performance of the solution, the experiments from [12] were reproduced and adapted to measure the performance of our system. We used a 40-core KVM virtual machine with 32GB of RAM running Debian GNU/Linux provided by the DEDIS laboratory. The experiments are stored in the `benchmark` branch in the repository mentioned in Chapter 4.

As shown on 5.1, the DKG phase of our solution shows similar performance to TDH2. This is not surprising considering that the former is based on the latter. The change of curve group has a negligible impact at most.

When it comes to decrypting a single message, we can see on figure 5.2 that IBE is significantly slower than TDH2. This was run on a single message and does not take into account the fact that one IBE round would likely cover multiple transactions, or any batching techniques that can be applied to TDH2. [12, p.14] One possible reason for this discrepancy is that curves that support bilinear mapping operations are more computationally expensive. [13]

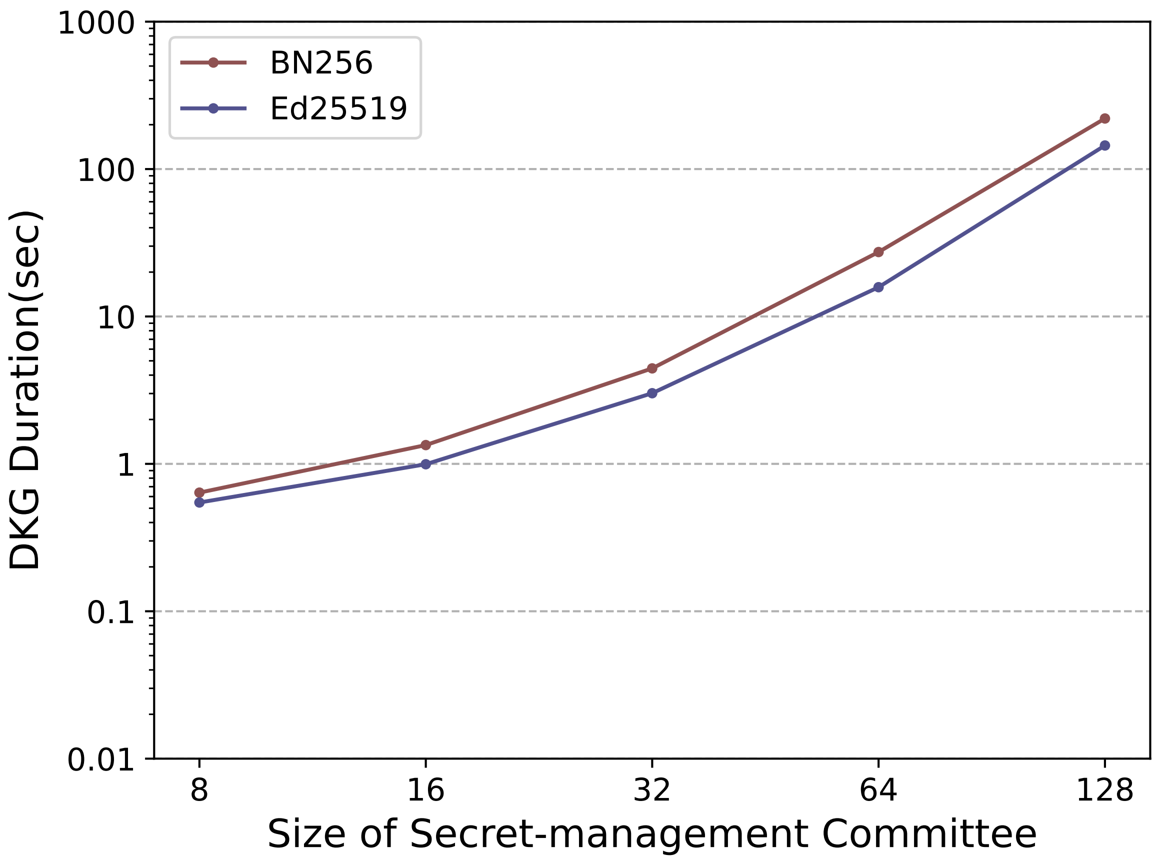

Figure 5.1: Performance of the DKG procedure on different curve groups

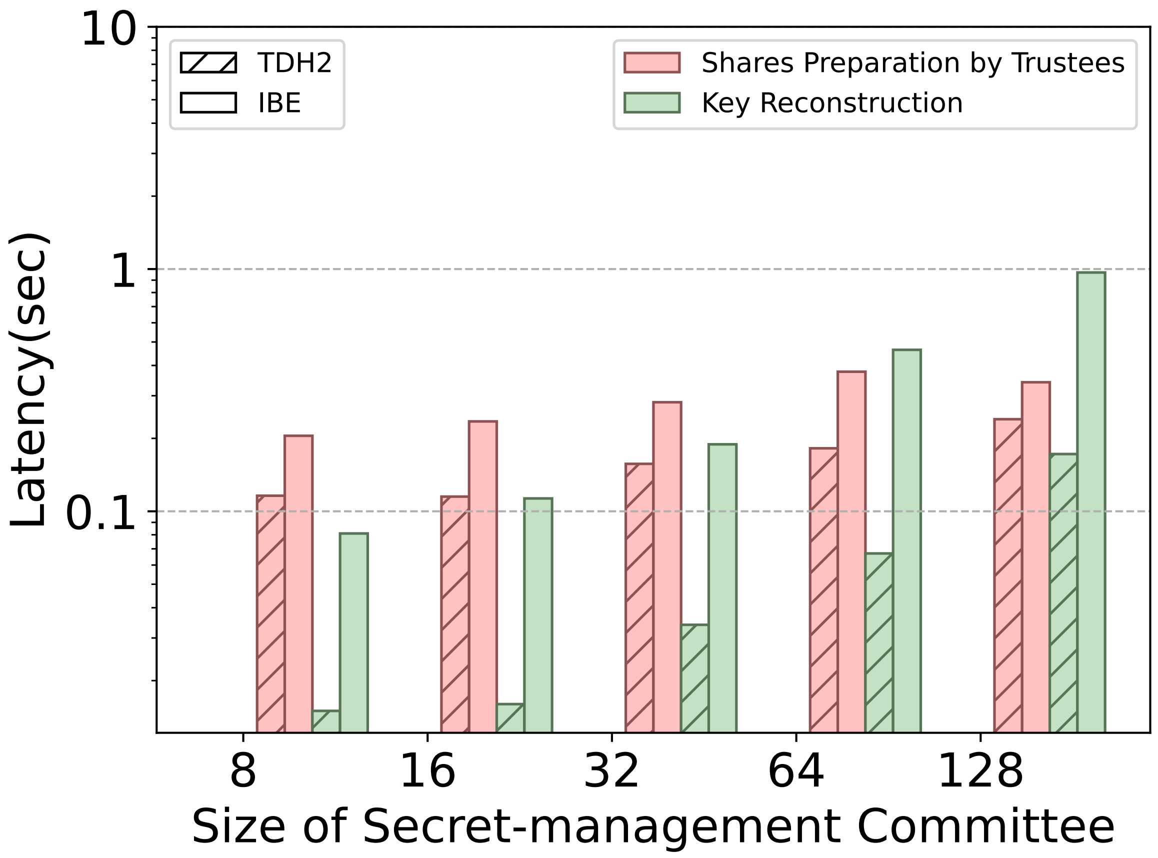

Figure 5.2: Performance of IBE compared to TDH2 for one message

Chapter 6

Discussion

Overall, we found that the F3B IBE protocol performs worse in a lab setting than its current rival. We also produced an implementation of the protocol that fits within the existing framework and can be further extended and studied. In particular, future work could involve connecting it with an Ethereum chain, as Wang has done. [11] Some tweaks such as using different elliptic curve groups could also be investigated.

As it stands, the IBE protocol makes the ciphertext longer than the plaintext by 128 bytes. This is the length of an uncompressed BN256 $\mathbb{G}_2$ point. This is not ideal since every additional byte on a transaction has to be stored and transmitted many times over. We suggest that this can be reduced with clever choices of groups. In particular, elements in BLS12-381's $\mathbb{G}_1$ can be stored in only 48 bytes.

Due to the performance impact, we might want to look for a faster IBE scheme. The scheme presented by Zheng, Zhou, and Cui does not require a pairing, and thus may be suitable. [13]

Coming back to the current F3B IBE protocol, a known and inherent weakness of it is that transactions can be decrypted even if they don't make it into the intended block. [12, p. 6] This may be partially or totally remedied by having a delay between block commitment and key reveal so that, if the transaction makes it to the next few blocks, it is still safe. This raises the question of how much it costs any given actor to delay inclusion for a target transaction in order to frontrun it, which depends heavily on the design of the underlying blockchain.

Chapter 7

Conclusion

In this work, we described the implementation of an IBE variant of the F3B protocol and its evaluation. We found that pairing-based cryptography delivered a conceptually simpler scheme, albeit at the cost of higher latency and computational costs. This lead us to propose multiple avenues for future research and development of the IBE approach to frontrunning protection.

Acronyms

CG	Consensus Group
DKG	Distributed Key Generation
IBE	Identity-based Encryption
SMC	Secret Management Committee

Bibliography

[1] Uniswap Labs □, @Uniswap. *Last month, Uniswap interface users hit an ATH since v3 launch in May '21. Thanks for choosing Uniswap as your go-to place to swap tokens □ pic.twitter.com/5GaaCNdxUs*. May 31, 2023. URL: `https://twitter.com/Uniswap/status/1663907328280653824`.

[2] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. *Constructing Elliptic Curves with Prescribed Embedding Degrees*. Cryptology ePrint Archive, Paper 2002/088. `https://eprint.iacr.org/2002/088`. 2002. URL: `https://eprint.iacr.org/2002/088`.

[3] Paulo S. L. M. Barreto and Michael Naehrig. *Pairing-Friendly Elliptic Curves of Prime Order*. Cryptology ePrint Archive, Paper 2005/133. `https://eprint.iacr.org/2005/133`. 2005. URL: `https://eprint.iacr.org/2005/133`.

[4] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-speed high-security signatures". In: *Journal of Cryptographic Engineering* 2 (2011), pp. 77–89.

[5] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. *Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges*. 2019. arXiv: `1904.05234 [cs.CR]`.

[6] Nicolas Gailly, Kelsey Melissaris, and Yolan Romailler. *tlock: Practical Timelock Encryption from Threshold BLS*. Cryptology ePrint Archive, Paper 2023/189. `https://eprint.iacr.org/2023/189`. 2023. URL: `https://eprint.iacr.org/2023/189`.

[7] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". In: *Journal of Cryptology* 20 (1999), pp. 51–83.

[8] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: 2015-07-01. 2008. URL: `https://bitcoin.org/bitcoin.pdf`.

[9] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *Advances in Cryptology*. Ed. by George Robert Blakley and David Chaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 47–53. ISBN: 978-3-540-39568-3.

[10] Victor Shoup and Rosario Gennaro. "Securing Threshold Cryptosystems against Chosen Ciphertext Attack". In: *J. Cryptology* 15 (2002), pp. 75–96. DOI: `10.1007/s00145-001-0020-9`.

[11] Shufan Wang. “Execution Layer Based Front-running Protection on Ethereum”. Available at `https://www.epfl.ch/labs/dedis/wp-content/uploads/2023/01/report-2022-3-ShufanWang-FrontRunningProtection.pdf`. Master’s thesis. École Polytechnique Fédérale de Lausanne, 2023.

[12] Haoqian Zhang, Louis-Henri Merino, Mahsa Bastankhah, Vero Estrada-Galinanes, and Bryan Ford. *F3B: A Low-Overhead Blockchain Architecture with Per-Transaction Front-Running Protection.* 2023. arXiv: `2205.08529 [cs.CR]`.

[13] Minghui Zheng, Huihua Zhou, and Guohua Cui. “An Improved Identity-Based Encryption Scheme Without Bilinear Map”. In: *2009 International Conference on Multimedia Information Networking and Security.* Vol. 1. 2009, pp. 374–377. DOI: `10.1109/MINES.2009.171`.