Demo Abstract: Thetacrypt – A Distributed Service for Threshold Cryptography On-Demand

Orestis Alpos
University of Bern
Bern, Switzerland
orestis.alpos@unibe.ch

Mariarosaria Barbaraci
University of Bern
Bern, Switzerland
mariarosaria.barbaraci@unibe.ch

Christian Cachin
University of Bern
Bern, Switzerland
christian.cachin@unibe.ch

Noah Schmid
University of Bern
Bern, Switzerland
noah.schmid@unibe.ch

Michael Senn
University of Bern
Bern, Switzerland
michael.senn@unibe.ch

ABSTRACT

With decentralized systems becoming prominent, often involving financial applications, several companies started looking for stronger security guarantees that could fit the new model. Threshold cryptography is being proposed as a solution for several central issues in the blockchain space, such as MEV prevention, wallet-key management, and randomness generation. The absence of a commonly available library for threshold cryptography is a major concern, as demonstrated by the recent NIST project [10]. In this paper, we present Thetacrypt, the first distributed service that aims at providing threshold cryptography as a service implementing the most requested threshold schemes.

CCS CONCEPTS

• **Computer systems organization** → **Fault-tolerant network topologies**; • **Theory of computation** → **Cryptographic protocols**; • **Security and privacy** → **Distributed systems security**.

KEYWORDS

Threshold cryptosystems, distributed algorithms.

ACM Reference Format:
Orestis Alpos, Mariarosaria Barbaraci, Christian Cachin, Noah Schmid, and Michael Senn. 2023. Demo Abstract: Thetacrypt – A Distributed Service for Threshold Cryptography On-Demand. In *Middleware Demos, Posters and Doctoral Symposium '23: Proceedings of the 24rd International Middleware Conference Demos, Posters and Doctoral Symposium (Middleware Demos, Posters and Doctoral Symposium '23), December 11–15, 2023, Bologna, Italy*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626564.3629100

1 INTRODUCTION

Distributing cryptographic capabilities among a cluster of nodes goes back to the work of Desmedt [5], who first exploited *secret sharing* [12] for the construction of a public-key cryptosystem such that only the collaboration of sufficiently many parties could lead to the cryptographic result. Such schemes are called *threshold cryptosystems*. Given a fixed number of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, any $(t+1)$ out of n shareholders of the secret key can successfully carry out a cryptographic operation, while t or fewer will not learn anything about the shared secret. Threshold cryptography distributes a cryptosystem among multiple parties and thereby increases its resilience. However, the intrinsic distributed nature of such protocols and the need to deal with complex infrastructure have limited their practical deployment so far.

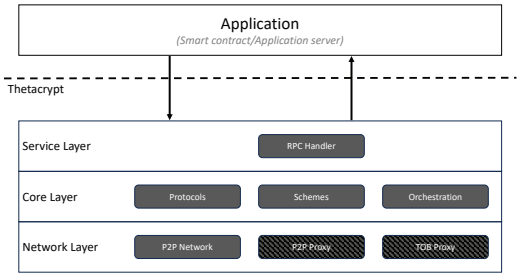

Figure 1: Thetacrypt architecture

Several implementations of threshold cryptosystems have been presented recently in targeted applications, especially for managing cryptocurrency wallets. [6, 9]. Others run as part of a distributed network, e.g., DFINITY incorporates threshold cryptography into the Internet Computer [4]. DRAND is a distributed randomness beacon producing verifiable and unbiased random numbers using distributed cryptography [7]. Partisia Blockchain provides private ("zero-knowledge") computation as a service [11]. Furthermore, protocols for building distributed cryptosystems are an active area of research in cryptography and blockchain systems, but very few of the more complex ones have actually been implemented or even deployed in practice.

This work aims at making threshold cryptography modular and at simplifying its deployment, such that the cryptography becomes independent of particular applications and can easily be composed with and integrated into diverse distributed protocols. The tension between (static) cryptographic algorithms and (dynamic) distributed algorithms makes threshold cryptosystems inherently complex to implement. We believe that this has hampered their widespread deployment so far.

We present Thetacrypt, a distributed service that provides threshold cryptography on demand. With its modular API, it can easily

Middleware Demos, Posters and Doctoral Symposium '23, December 11–15, 2023, Bologna, Italy

ACM ISBN 979-8-4007-0429-1/23/12.
https://doi.org/10.1145/3626564.3629100

integrate threshold cryptographic algorithms in diverse distributed applications. Making threshold cryptography a separate component in distributed applications will improve the overall security and offer higher flexibility for application developers. Thetacrypt provides the software infrastructure to execute such protocols, takes care of the challenges related to a correct implementation of the cryptographic primitives, can handle complex multi-round distributed schemes, and offers a modular way to deploy and integrate distributed cryptosystems. To our knowledge, Thetacrypt is the first project that aims at providing threshold cryptography as a service that can be integrated in a flexible way with multiple applications.

2 OVERVIEW

Thetacrypt operates as a distributed service and requires a network of *nodes*, each running a *Thetacrypt instance* in a dedicated process. Thus, the service instance is flexible, can maintain state, and follows the lifecycle of its target application. Whereas the application invokes the service through an RPC request establishing a *client-server* interaction, the Thetacrypt instances rely on a peer-to-peer network for *internode coordination*.

Thetacrypt encompasses three layers (Fig. 1): the *service layer*, presenting the management code to handle the service itself and exposing an RPC endpoint; the *core layer*, orchestrating and scheduling concurrent protocol executions performing a cryptographic operation; the *network layer*, handling the communication among the nodes through a peer-to-peer (p2p) network. The latter additionally provides two proxy modules that enable the integration with an existing replicated service and its network layer.

The core of the service is a *schemes module*, which provides the cryptographic primitives for the threshold schemes and powers the core layer. This module is self-contained and could be used alone. Table 1 presents an overview of the implemented cryptographic schemes.

An additional binary *ThetacryptCLI*, is available to the admin to perform tasks, such as the initial setup of the network with a trusted dealer, encryption under a specific public key and verification of signatures returned by the service.

3 CONFIGURATIONS

The easiest way to deploy Thetacrypt is to run it as a dedicated distributed service, coupled with a suitable application layer. The application is then responsible for ensuring that every Thetacrypt instance receives valid requests and that they respect the particular ordering requirements of the threshold scheme by implementing this logic in the application.

Scheme type	Cryptographic problem	Verification strategy	Reference
Th. Coin	DL	ZKP	[3]
Th. Signature	RSA	ZKP	[13]
Th. Signature	DL	ZKP	[8]
Th. Signature	DL	pairings	[2]
Th. Cipher	DL	ZKP	[14]
Th. Cipher	DL	pairings	[1]
Th. VRF	DL	pairings	[2]

Table 1: Implemented threshold schemes

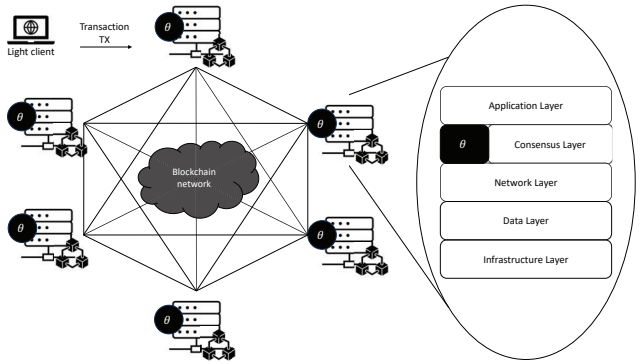

Figure 2: Thetacrypt's deployment over a blockchain

The intended deployment mode of Thetacrypt, however, is an integration with a replicated service, i.e., a network that uses the state-machine replication paradigm. In this setting, each node executes code deterministically, in parallel to the other nodes, and invokes Thetacrypt for cryptographic operations. This deployment of Thetacrypt is suitable for integration with blockchain platforms. The application using Thetacrypt is actually a *smart contract* that accesses it through a specific API call on the *validator node* of the blockchain network. Figure 2 illustrates this deployment and where the Thetacrypt process resides in the blockchain stack. In this setting, the p2p network used by Thetacrypt internally extends through a *p2p proxy module* to the blockchain network for internode communication. Some cryptographic schemes (like distributed key generation) require totally ordered communication, i.e., consensus on the order of messages. This is provided by accessing the underlying blockchain as a total order broadcast channel through a *TOB proxy module* (Fig. 1).

REFERENCES

[1] Joonsang Baek and Yuliang Zheng. 2003. Simple and efficient threshold cryptosystem from the Gap Diffie-Hellman group. In *GLOBECOM*. IEEE, 1491–1495.
[2] Dan Boneh, Ben Lynn, and Hovav Shacham. 2004. Short Signatures from the Weil Pairing. *J. Cryptol.* 17, 4 (2004), 297–319.
[3] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography. *J. Cryptol.* 18, 3 (2005), 219–246.
[4] Jan Camenisch, Manu Drijvers, Timo Hanke, Yvonne-Anne Pignolet, Victor Shoup, and Dominic Williams. 2022. Internet Computer Consensus. In *PODC*. ACM, 81–91.
[5] Yvo Desmedt. 1994. Threshold cryptography. *Eur. Trans. Telecommun.* 5, 4 (1994), 449–458.
[6] Dfns 2023. Dfns: Web3 wallets as an API. https://www.dfns.co.
[7] drand 2023. drand: Distributed randomness beacon. https://drand.love.
[8] Chelsea Komlo and Ian Goldberg. 2020. FROST: Flexible Round-Optimized Schnorr Threshold Signatures. In *SAC (Lecture Notes in Computer Science, Vol. 12804)*. Springer, 34–65.
[9] Yehuda Lindell. 2023. Digital Asset Management with MPC. https://www.coinbase.com/blog/digital-asset-management-with-mpc-whitepaper.
[10] NIST. 2019. Multi-Party Threshold Cryptography (MPTC) Project. https://csrc.nist.gov/Projects/Threshold-Cryptography/.
[11] Partisia Blockchain. 2023. Partisia: Solving the Blockchain Trilemma. https://partisiablockchain.com.
[12] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979), 612–613.
[13] Victor Shoup. 2000. Practical Threshold Signatures. In *EUROCRYPT (Lecture Notes in Computer Science, Vol. 1807)*. Springer, 207–220.
[14] Victor Shoup and Rosario Gennaro. 2002. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. *J. Cryptol.* 15, 2 (2002), 75–96.