Grant Agreement Number: 825225

Safe-DEED

www.safe-deed.eu

PSI/MPC and Multi-User Data Aggregation v1/2

Deliverable number	*D5.3*
Dissemination level	*Public*
Delivery data	*due 29.11.2019*
Status	*Final*
Authors	*Lukas Helminger, Roman Walch*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825225.

Executive Summary

The vast possibilities of multi-party computation were already shown in theory. It is now time for the next step. That means to develop efficient cryptographic primitives and, further down the road, real-world applications. The end goal is to aid and encourage the development and use of privacy-enhancing technologies. For example, these technologies will help companies to perform their data analysis in accordance with the General Data Protection Regulation.

In the task contributing to this deliverable, we build a new cryptographic primitive using multi-party computation called MPC-accumulators. Thereby, we showed that it is possible to gain performance through distributed trust - most noticeably as an application to certificate transparency. More concretely, MPC-accumulators can help to ensure that the browsing history of users is not revealed when verifying the authenticity of certificates.

Further, we describe two major next steps - first, the development of efficient real-world applications, where we will focus on private set intersection protocols. Today PSI protocols are either fast and lack security or are secure but not efficient enough for a practical purpose. We try to find a better trade-off between security and runtime. Finally, we want to use our experience to build tools for the development of MPC applications. These tools will be provided to the software community and therefore enable more people to write private-enhancing programs.

Table of Contents

1 Introduction

This document aims to explain the research progress we made in the topic of multi-party computation (MPC) (In D5.1, we gave an introduction to MPC and more generally the requirements for secure computations on large datasets with multiple owners.) More concretely, through this deliverable, we want to enable security experts and software engineers to understand our research paper better. This paper is about a new cryptographic primitive, so-called multi-party computation accumulators (MPC-accumulators) in their possible applications. Developing this new primitive involved a little contribution to the secure computation framework FRESCO [1].

1.1 Research output

An MPC-accumulator is a new basic cryptographic primitive. We described this new primitive with all technical details in the research paper *Multi-Party Public-Key Accumulators: Performance Through Distributed Trust*. Since the paper is currently in the peer-review process of a major conference, we can only provide an extended abstract. However, as soon as this process is finished, the paper will be online available. Two Safe-DEED researchers contributed to this paper. Our goal in this document is to make the paper accessible to a wider audience. Therefore, we describe the main ideas in a slightly higher level than in the original research paper. The functionality of MPC-accumulators best can be seen by an example. More precisely, we will explain how MPC-accumulators can be used to make secure communication more private. To follow the arguments of the application, we have to explain the idea of certificate-transparency 2. After this short excursion, we directly look at MPC-accumulators in Section 3. For a technical rigor description of MPC-accumulators, including security proofs, we refer the interested reader to research paper.

1.2 Contributing to the community

For the implementation of MPC-accumulators, we had to extend the current functionality of FRESCO (see deliverable D5.5). FRESCO's aims to enable software engineers to do fast prototyping of MPC protocols without the necessity of having a substantial cryptographic background. Already the EU Horizon 2020 project SODA[1] contributed to the development of FRESCO. Safe-DEED is also interested in improving MPC frameworks. Therefore we will have a more in-depth look at how we can best integrate our ideas for improvement into FRESCO. In this way, our results are easily usable by the whole community.

1.3 New Security Model

In addition, to the above efforts we are currently investigating a rather new security model called public verifiability with covert security (PVC) [2]. PVC is a security model for two-party computation that lies in-between semi-honest and malicious security (see Deliverable 5.1). At Eurocrypt 2019, Hong et al. [6] presented the first efficient protocol for PVC. The performance of their protocol is nearly as good as the performance of semi-honest protocols,

[1] `https://www.soda-project.eu/`

while simultaneously offering almost malicious security, at least in terms of practical security. In other words, it has the advantages of both traditional security models - strong security and fast runtime. As far as we know, there is only the proof of concept from Hong et al., but no real-world application. We are currently trying to apply PVC to PSI. If we can reproduce the good results form Hong et al., the new PSI protocol could be the best choice for many practical applications. We are still at an early stage but optimistic about gaining lot of progress in the next months.

2 Certificate-Transparency

In order to understand the application of MPC-accumulators to certificate-transparency (CT), we will first look at today's usage of TLS in Section 2.1. In Section 2.2 we then focus on the definition of the public-key infrastructure (PKI) and highlight its weakness with respect to certificates. After that we look at certificate transparency in Section 2.3. This is a promising approach for more trustworthy certificates. However, as we look deeper into how CT is implemented in practice, we will discover a privacy flaw.

2.1 Transport Layer Security

These days Transport Layer Security (TLS) [13] is the leading cryptographic protocol for secure communication over the internet. To see this, one has to know that HTTPS relies on TLS. A recent statistic from Let's Encrypt on the usage of HTTPS shows that around 80% of webpages loaded by the browser Firefox use HTTPS[2]. That means an improvement in TLS affects many users and applications.

One crucial task of TLS is that it enables a client and a server to agree on a common secret key. This shared secret can then be used to encrypt the actual communication. This essential part of TLS is called the key agreement. The key agreement is usually done with the help of public-key cryptography. Real-world public-key cryptography is dependent on a so-called public-key infrastructure (PKI).

2.2 Public-Key Infrastructure

Simply put, the PKI tells one which public key belongs to whom. More specifically, the non-profit standard organization Internet Engineering Task Force (IETF) defined the PKI as "the set of systems, policies, and procedures used to protect the confidentiality, integrity, and authenticity of communications between Web browsers and Web content servers"[3].

Unfortunately, the indispensable PKI is not flawless. One of its shortcomings is its lack of transparency and the higher possibility of compromised certificates that this entails. Compromised certificates had lead to far-reaching and spectacular security breaches, e.g., the 2011 hack of the Dutch certificate authority (CA) DigiNotar[4] was exploited for a man-in-the-middle

[2] https://letsencrypt.org/de/stats/

[3] https://datatracker.ietf.org/doc/charter-ietf-wpkops/

[4] https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html

attack against Google Services from unknown entities in Iran [12]. After that incident, there were considerable efforts to lower the probability of compromised certificates.

2.3 Certificate Transparency

A promising countermeasure to tackle the above-described issue is Certificate-Transparency (CT) [8, 9]. This framework acts as an extension of the existing PKI and is already used by Google's browser chrome for certificates issued after April 2018[5]. A high-level overview of CT can be found in [5]. A detailed technical description of the framework is out of scope of this deliverable and also not necessary to follow the document. However, we refer the interested reader to the document done by Google[6]. In the following paragraphs, we will give a short overview of the structure and the functionality of CT. In the description of CT, we do not aim for completeness, but rather focus only on points relevant to understand the application of MPC-accumulators.

2.3.1 Structure and Functionality

One must not understand the CT as an alternative to the PKI. In contrast, it is an extension of the current PKI without much integration effort. The CT framework consists of three main parts: the logs, the monitors, and the auditors. These three parts work together to counter the problem of compromised certificates.

Log servers manage a list of certificates. In addition to storing the certificates, the log is able to produce a proof that a given certificate is contained in the log. To ensure the integrity of such a membership test over time, one can also request a consistency proof. That means that the log is an append-only list. These log servers are the backbone of CT because every certificate that is not contained in a log is considered to be compromised.

Monitors are responsible for watching the log servers. They do this by keeping copies of the logs. On the one hand, they look for suspicious permissions and on the other hand, check the consistency of the log over time. If the monitors detect a potential misuse of a certificate, or duplicate certificate, they can inform the domain owners.

Auditors are the authority that can check if a given certificate is contained in a log. That means they can request a membership proof of a specific certificate from the log servers. The auditors may, like the monitors, verify the consistency of logs with respect to changes over time.

2.3.2 Realization

Conceptually, those three roles and their functionality can be formalized by cryptographic accumulators (see [4] for an overview). A cryptographic accumulator is usually implemented by Merkle-trees [11] as it is also the case for CT. Unfortunately, the current implementation of CT is not fully privacy-friendly. Every time a client requests membership proof of a particular certificate, she implicitly reveals her browsing behavior to the log server. This could, for example, lead to the following scenario. Politically exposed persons would opt-out of the CT ecosystem, because they do not want to be associated with particular sites. In turn, such behavior would

[5] https://www.section.io/blog/chrome-ct-compliance/
[6] https://www.certificate-transparency.org/

result in a higher probability of being a target for adversaries. To sum up, through the use of the CT framework, the likelihood of getting a compromised certificate is drastically reduced. However, there is still room for improvement on the privacy of clients.

3 Multi-Party Computation Accumulators

Several approaches have been proposed to solve the privacy issues of CT. We will focus on privacy-preserving retrieval of membership proofs from CT log servers. The following section is organized in the following way. In Section 3.1, we take a look at how accumulators are currently used in CT to guarantee privacy. All of these accumulators are based on Merkle-trees. However, there is an exciting alternative, which we will discuss in Section 3.2. Finally, in Section 3.3, we will come to multi-party computation accumulators, our main contribution to this version of the deliverable.

3.1 Accumulators in CT

Most of the current solutions for privacy-friendly CT logs do not sufficiently scale. This scalability is desperately needed to deal with the ever-growing number of certificates[7]. One of the most fruitful approaches to solve the privacy flaw described in Section 2.3.2 is from Lueks and Goldberg [10]. It is based on a private information retrieval (PIR) protocol [3]. PIR is a cryptographic primitive, allowing a client to retrieve an item from a server database. The server does not learn which item the client retrieved. In particular, a PIR can be used to verify the membership of a given certificate in the log without the log servers learning anything about the requested certificate. More specifically, Lueks and Goldberg use multi-server PIR for computational and communication efficiency. Even with this multi-server set up, their construction lacks scalability.

Based on the above-described approach Kales et al. [7] presented a design that can handle a huge number of certificates. Their design is also relying on Merkle-trees, even though whenever a new certificate is added to the log, the Merkle-tree has to be recomputed. That means Merkle-trees lack an efficient update functionality. Kales et al. also give reasons why they chose Merkle-trees over the theoretically more suitable dynamic public-key accumulators. To follow their argumentation, we take a brief look at dynamic public-key accumulators.

3.2 Dynamic Public-Key Accumulators

Dynamic public-key accumulators have the same basic functionality as Merkle-trees. In addition, they allow to dynamically update the accumulator without recomputing the whole value. The last property makes dynamic public-key accumulators an ideal fit for the desired functionality in CT. As in every public-key primitive, there is a key pair. It consists of a private and a public key. For a dynamic public-key accumulator to be efficiently computable, knowledge of the private key - also called secret trapdoor - is required. The knowledge of the secret trapdoor would allow a malicious log server to produce membership proofs for every certificate, even if it was not contained in the log. That means the client has no way to check a given certificate

[7] https://ct.cloudflare.com/

properly. On the other hand, if the log server does not have access to the secret trapdoor, then dynamic public-key accumulators are too slow. This was recently observed by Kales et al. and the reason why they used Merkle-trees.

3.3 MPC-Accumulators

From now on all the postulated results can be found in the full version of the appended paper except otherwise mentioned. Recall that the protocols from Lueks and Goldberg and Kales et al. already require a multi-server setting. Therefore it is only naturally to use MPC (for an overview, see deliverable D5.1). MPC enables us to solve the dilemma from the last paragraph. The secret trapdoor is split up between the servers, i.e., every server only gets a share of the private key. The private key can only be reconstructed with all the shares. That means, as long as one server stays uncorrupted, no proof of membership can be forged. On the other side, the log servers can use the shared secret trapdoor to do computations efficiently. All those computations are done by MPC protocols. Therefore we called this new construction multi-party public-key accumulators.

3.3.1 Construction

Based on the just presented abstract idea, we were able to come up with two concrete maliciously-secure protocols. The first one is based on the oldest public-key encryption scheme RSA, whereas the second one is based on bilinear pairings. Bilinear pairings and their security (t-SDH) are closely related to elliptic curve cryptography and, therefore, well studied. Existing dynamic public-key accumulators based on bilinear pairings (called t-SDH accumulators) turned out to be highly suitable for adaption through MPC. We called them MPC-t-SDH accumulators.

3.3.2 Advantages

Switching the accumulator from Merkle-trees to our MPC-t-SDH accumulator would result in constant size and small membership proofs. More concretely, the size of a membership proof for a given certificate would reduce by a factor of 10. In other words, the overhead of a membership proof to a typical DER-encoded X509 certificate used in TLS would decrease from $25-50\%$ to $2.5-5\%$. Further, the estimated total communication between a client and the log servers would be reduced by a factor of 3.5.

3.3.3 Implemenation

We implemented the proposed MPC-t-SDH accumulator and evaluated it against small to large sets. Detailed results can be read out from the paper (full version). Our implementation is open-source and will be put online as soon as the review process for the corresponding paper is finished. More details about the internals of the implementation can be found as part of deliverable D5.5 There, we also included a step-by-step manual on how to download, install, and edit the source code.

4 Conclusion

The main contribution to this version of the deliverable is the research output. More concretely, the submitted paper in the appendix. To make it accessible to a broader audience, we explained MPC-accumulators in a high-level manner. As an example of usage of MPC-accumulator we looked at certificate-transparency. For future work, we have two main goals at the moment. On the on hand, we try to develop a PSI protocol that has public verifiability with covert security. This would probably lead to a PSI protocol with the best security performance trade-off. On the other hand, we want to actively contribute to a secure computation framework, namely FRESCO. Through such frameworks implementing MPC protocols should get easier for the software community and therefore result in more privacy-friendly applications.

5 References

[1] Alexandra Institute. FRESCO - a FRamework for Efficient Secure COmputation, 2019.

[2] Gilad Asharov and Claudio Orlandi. Calling out cheaters: Covert security with public verifiability. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 681–698. Springer, 2012.

[3] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50. IEEE, 1995.

[4] David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In *CT-RSA*, volume 9048 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 2015.

[5] Lukas Gebhard. Shedding light on certificates: The web pki and certificate transparency. 07 2018.

[6] Cheng Hong, Jonathan Katz, Vladimir Kolesnikov, Wen-jie Lu, and Xiao Wang. Covert security with public verifiability: Faster, leaner, and simpler. In *EUROCRYPT (3)*, volume 11478 of *Lecture Notes in Computer Science*, pages 97–121. Springer, 2019.

[7] Daniel Kales, Olamide Omolola, and Sebastian Ramacher. Revisiting user privacy for certificate transparency. In *EuroS&P*, pages 432–447. IEEE, 2019.

[8] Ben Laurie. Certificate transparency. *ACM Queue*, 12(8):10–19, 2014.

[9] Ben Laurie, Adam Langley, and Emilia Käsper. Certificate transparency. *RFC*, 6962:1–27, 2013.

[10] Wouter Lueks and Ian Goldberg. Sublinear scaling for multi-client private information retrieval. In *Financial Cryptography*, volume 8975 of *Lecture Notes in Computer Science*, pages 168–186. Springer, 2015.

[11] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York.

[12] J Ronald Prins and Business Unit Cybercrime. Diginotar certificate authority breach'operation black tulip'. *Fox-IT, November*, page 18, 2011.

[13] Eric Rescorla. The transport layer security (TLS) protocol version 1.3. *RFC*, 8446:1–160, 2018.

A Paper

Multi-Party Public-Key Accumulators: Performance Through Distributed Trust

Lukas Helminger[1,2], Daniel Kales[1], Sebastian Ramacher[3], and Roman Walch[1,2]

[1] Graz University of Technology, Graz, Austria
[2] Know-Center GmbH, Research Center for Data-Driven Business & Big DataAnalytics, Austria
[3] AIT Austrian Insititute of Technology, Vienna, Austria

Abstract. Accumulators provide compact representations of large sets and enjoy compact membership witnesses. Besides constant-size witnesses, public-key accumulators provide efficient updates of both the accumulator itself and the witness; however, they come with two drawbacks: they require a trusted setup and – without knowledge of the secret trapdoors – their performance is not practical for real-world applications with large sets. Recent improvements in the area of secure multi-party computation allow us to replace the trusted setup with a distributed generation of the public parameters. We present versions of the dynamic public-key accumulators giving access to the more efficient witness generation, and update algorithms using the shares of the secret trapdoors sampled by the parties generating the public parameters.
In this paper, we introduce multi-party public-key accumulators dubbed dynamic linear secret-shared accumulators. Specifically, for the t-SDH-based accumulators, we provide a maliciously-secure variant sped up by a secure multi-party computation (MPC) protocol (IMACC'19) built on SPDZ. For this scheme, a proof-of-concept implementation is provided, which substantiates the practicability of public-key accumulators in this setting. With our implementation in the FRESCO MPC framework, we obtain more efficient accumulators for both medium-sized (2^{10}) and large (2^{14} and above) accumulated sets.
For RSA-based accumulators, we discuss possibilities to instantiate the accumulator in the MPC setting based on recent work on the multi-party RSA key generation (CRYPTO'18). This construction is mostly of theoretical interest since the underlying MPC protocol is not efficient enough to handle many accumulator queries and some of the involved protocols leak a small number of bits of the secret trapdoor.

Keywords: multiparty computation, dynamic accumulators, distributed trust

1 Introduction

A cryptographic accumulator [10] allows one to accumulate a finite set $\mathcal{X}$ into a succinct value called the accumulator. For every element in the accumulated

set, one can efficiently compute a witness certifying its membership. However, it should be computationally infeasible to find a witness for non-accumulated values. Accumulators are an important primitive and building block in many cryptographic protocols. In particular, Merkle trees [37] have seen many applications as accumulators in both the cryptographic literature but also in practice. For example, they have been used to implement Certificate Transparency (CT) [31,32,22] where all issued certificates are publicly logged, i.e., accumulated, and then clients can check that the server's certificate is included in a log by requesting witnesses from the log servers. Additionally, cryptocurrencies such as Bitcoin heavily rely on Merkle trees as well. Accumulators in general also find application in redactable signatures [44,20], anonymous credentials [12], ring, and group signatures [33,21,28], membership revocation [4], anonymous cash [38], authenticated hash tables [43], among many others.

Interestingly, when looking at applications of accumulators deployed in practice, many systems rely on Merkle trees. Most prominently we can observe this fact in CT. Even though new certificates are continuously added to the log, the system is designed around a Merkle tree that gets recomputed all the time instead of updating a dynamic public-key accumulator. The reason is two-fold: first, for dynamic accumulators to be efficiently computable, knowledge of the secret trapdoor used to generate the public parameters is required. Without this information, witness generation and accumulator updates are simply too slow for large sets (as recently observed in [27]). Secondly, in this setting it is of paramount importance that the log servers do not have access to the secret trapdoor. Otherwise malicious servers would be able to present membership witnesses for each and every certificate even if it was not included in the log.

The latter issue can also be observed in other applications of public-key accumulators. Zerocoin [38], for example, employs RSA accumulators together with Pedersen commitments and signatures of knowledge to provide an extension to Bitcoin for fully anonymous transactions. In short, when spending a coin, all coins are accumulated, and a witness for the membership of the spent coin is produced. The accumulator has to be re-computed during both the spending operation and during verification. The protocol, however, has to rely on the fact that during the generation of the parameters of the RSA accumulator, the secret trapdoor information has been deleted. Otherwise, it is possible for an adversary to spend more coins than have been minted.

In these types of protocols, the generation of the public parameters is problematic. In fact, it requires to put significant trust in the parties generating those parameters. If they would act maliciously and not delete the secret trapdoors, they would be able to break all these protocols in one way or another. To circumvent this problem, Sander [45] proposed a variant of an RSA-based accumulator from RSA moduli with unknown factorization. Alternatively, secure multi-party computation (MPC) protocols make it possible to compute the public parameters and thereby replace the trusted third party. As long as a large enough subset of parties participating is honest, the secret trapdoor is not available to malicious parties. Over the years, efficient solutions for distributed parameter

generation have emerged, e.g., for distributed RSA key generation [23], or distributed ECDSA key generation [34]. One very prominent and wildly publicized example of such an approach is the "ceremony" executed for Zcash, where an MPC protocol involving hundreds of participants was used to generate the public parameters for the proof system [11].

Based on the recent progress in efficient MPC protocols, we ask the following question: *what if the parties kept their shares of the secret trapdoor?* Are the algorithms of the public-key accumulators exploiting knowledge of the secret trapdoor faster if performed within an (maliciously-secure) MPC protocol than their variants relying only on the public parameters?

1.1 Our Contribution

In this work we tackle this question and give a positive answer for discrete logarithm-based accumulators. Our contributions can be summarized as follows:

- Based on recent improvements on distributed key generation both of RSA moduli and discrete logarithms, we provide the first dynamic public-key accumulators without trusted setup. During the parameter generation, the involved parties keep their shares of the secret trapdoor. Consequently, we can provide MPC protocols secure in the semi-honest and the malicious security model, respectively, implementing the algorithms for accumulator generation, witness generation, and accumulator updates exploiting the shares of the secret trapdoor. To the best of our knowledge, this is the first work that uses secure multiparty computation to build distributed cryptographic accumulators and in particular, linear secret-shared accumulators.
- We provide a maliciously-secure protocol for t-SDH accumulators, and especially for the dynamic accumulator due to Derler et al [19], which is based on the accumulator by Nguyen [41]. In particular, this protocol enables updates to the accumulator independent of the size of the accumulated set. For increased efficiency, we also transport this accumulator to the Type-3 pairing setting. Due to the structure of the bilinear groups setting, the construction nicely generalizes to any number of parties.
- We provide a proof-of-concept implementation of our protocols in FRESCO [2], a framework allowing fast prototyping of MPC protocols. We evaluate the efficiency of our protocols and compare them to the performance of an implementation, having no access to the secret trapdoors as usual for the public-key accumulators. We evaluate our maliciously-secure protocol in the LAN and WAN setting for various choices of parties and accumulator sizes. For the latter, we choose sizes up to 2^{14}. Specifically, for the t-SDH accumulator, we observe the expected $\mathcal{O}(1)$ runtimes for witness creation and accumulator updates, which cannot be achieved without access to the trapdoor. Notably, the MPC-enabled accumulator creation algorithms are faster in the LAN setting than its counterpart only taking the public parameters for up to 3 parties (2^{10} elements) to 5 parties (2^{14} elements), and we expect even greater improvements as the size of the accumulated set grows further.

– Additionally, we discuss the application of our ideas to the strong RSA accumulator due to [6]. However, the most efficient protocols currently available in the literature are leaky multiparty computation protocols and are only applicable to two parties. Hence, we only discuss this construction from a theoretical point of view.

On top of that, we discuss how our proposed MPC-based accumulators might impact applications like CT and the privacy-preserving extension based on private information retrieval (PIR) [27]. In particular, the size of the witnesses stored in certificates or sent as part of the TLS handshake is significantly reduced without running into performance issues.

1.2 Our Techniques

We give a short overview of how our construction works. Let us start with accumulator based on the t-SDH assumption. The construction is based on the fact that given powers $g^{s^i} \in \mathbb{G}$ for all i up to t where $s \in \mathbb{Z}_p$ is unknown, it is possible to evaluate polynomials $f \in \mathbb{Z}_p[X]$ up to degree t at s in the exponent, i.e., $g^{f(s)}$. To evaluate the polynomial in the exponent, one takes the coefficients of the polynomial, i.e., $f = \sum_{i=0}^{t} a_i X$, and computes $g^{f(s)}$ as

$$\prod_{i=0}^{t} \left(g^{s^i}\right)^{a_i}.$$

The idea of the accumulator is to then define a polynomial involving all elements of the set as roots. The accumulator is then comprised of that polynomial evaluated at s in the exponent. A witness is simply the corresponding factor canceled out, i.e., $g^{f(s)(s-x)^{-1}}$. Verification of the witness is performed by checking whether the corresponding factor and the witness match $g^{f(s)}$ using a pairing.

If s is known, all the computations are significantly more efficient: $f(s)$ can be directly evaluated in $\mathbb{Z}_p$ and then the generation of the accumulator only requires one exponentiation in $\mathbb{G}$. Similarly, computation of the witness also only requires one exponentiation in $\mathbb{G}$, since $(s-x)^{-1}$ can first be computed in $\mathbb{Z}_p$. For the latter, the asymptotic runtime is thereby reduced from $\mathcal{O}(|\mathcal{X}|)$ (i.e., linear in the number of accumulated elements) to $\mathcal{O}(1)$. But the improvement comes at a cost: if s is known, witnesses for non-members can be produced.

On the other hand, if s is first produced by multiple parties in an additively secret-shared fashion, these parties can cooperate in a secret-sharing based MPC protocol. Thereby, all the computations can still benefit from the knowledge of s. Indeed, the parties would compute their share of $g^{f(s)}$ and $g^{f(s)(s-x)^{-1}}$ respectively and thanks to the partial knowledge of s could still perform all operations – except the final exponentiation – in $\mathbb{Z}_p$.

Now let's discuss an accumulator based on RSA. The idea here is to multiply all elements of the accumulated set and raise some base element g to the power of the product modulo an RSA modulus $N = p \cdot q$, i.e.,

$$g^{\prod_{x \in \mathcal{X}} x} \mod N.$$

Similar to what is done for the t-SDH accumulator, the witness for an element is produced by canceling the corresponding factor from the product. Verification of the witness is again performed by checking whether the witness and the corresponding factor match the accumulator.

If the factorization of N is known, all computations can be done more efficiently. First of all, the Chinese remainder theorem can be used for a generic performance improvement. Secondly, if the factorization is available, it is possible to efficiently compute inverses modulo $\phi(N) = (p-1) \cdot (q-1)$. Thereby, instead of recomputing the product in the exponent to compute a witness, the accumulator just needs to be raised to the inverse of an element to obtain the witness. Again, the asymptotic runtime is reduced from $\mathcal{O}(|\mathcal{X}|)$ to $\mathcal{O}(1)$ with the same drawback that membership witnesses for non-members can be produced.

However, the inverse of some publicly known element needs to be computed during the key generation algorithm of typical RSA-based cryptosystems. Hence, the distributed RSA key generation algorithms such as the one by Frederiksen et al. [23] implement that functionality. There, two parties sample the primes p and q in a first step, where both parties obtain shares of the two primes. After the primes have been selected, the inverse of the public exponent is computed using the shares of the primes. Hence, if we split the first and the second step, we can leverage the shares of the primes to compute inverses to cancel terms from the accumulator to produce witnesses. Note though, that the protocol by Frederiksen et al. leaks secret bits similar to other distributed RSA protocols, but additionally trades the leakage of additional bits for better performance.

1.3 Related Work

When cryptographic protocols are deployed that require the setup of public parameters by a trusted third party, issues similar to those mentioned for public-key accumulators may arise. As discussed before, especially cryptocurrencies had to come up with ways to circumvent this problem for accumulators but also the common reference string (CRS) of zero-knowledge SNARKs [13]. Here, trust in the CRS is of paramount importance to prevent malicious provers from cheating. We note that there are alternative approaches, namely subversion-resilient zk-SNARKS [8] to reduce the trust required in the CRS generator. However, subversion-resilient soundness and zero-knowledge at the same time has been shown to be impossible by Bellare et al. Abdolmaleki et al. [1] provided a construction of zk-SNARKS, which was later improved by Fuchsbauer [24], achieving subversion zero-knowledge, by adding a verification algorithm for the CRS and then only the verifier needs to trust the correctness of the CRS. In the random oracle model (ROM), those considerations become less of a concern and the trust put into the CRS can be minimized, e.g., as done in the construction of STARKs [9].

Approaches that try to fix the issue directly in the formalization of accumulators and corresponding constructions have also been studied. For example, Lipmaa [36] proposed a modified model tailored to the hidden order group setting. In this model, the parameter setup is split into two algorithms, Setup and

Gen, whereas the adversary can control access the trapdoors output by Setup, but cannot influence nor access the randomness used by Gen. However, secure constructions in this model so far have been provided for rather unstudied assumptions based on modules over Euclidean rings, and are not applicable to the efficient standard constructions we are interested in.

The area of secure multiparty computation has seen a lot of interest both in improving the MPC protocols itself to a wide range of practical applications. In particular, SPDZ [18,16] has seen a lot of interest, improvements and extensions [29,30,15,42]. This interest also lead to multiple MPC frameworks, e.g. FRESCO [2], MP-SPDZ [47] and SCALE-MAMBA [3], enabling easy prototyping for researchers as well as developers. For practical applications of MPC, one can observe first MPC-based systems turned into products such as Unbound's virtual hardware security model (HSM).[4] For such a virtual HSM, one essentially wants to provide distributed key generation [23] together with threshold signatures [17] allowing to replace a physical HSM. Similar techniques are also interesting for securing wallets for the use in cryptocurrencies, where especially protocols for ECDSA [26,25,35] are of importance to secure the secret key material. Additionally, addressing privacy concerns in machine learning algorithms has become increasingly popular recently, with MPC protocols being one of the building blocks to achieve private classification and private model training [40,48,7,39]. Recent works [46] also started to generalize the algorithms that are used as parts of those protocols allowing group operations on elliptic curve groups with secret exponents or secret group elements.

We will work in the universal composability (UC) model [14] as common in the MPC literature. Only recently, the security of accumulators in the UC model has been investigated for the first time in [5]. Most importantly, existing constructions of cryptographic accumulators, and in particular the RSA and t-SDH accumulator, also satisfy the definitions of universally composable accumulators and are secure in the UC model.

Acknowledgments

This work was supported by EU's Horizon 2020 project Safe-DEED, grant agreement n°825225, and EU's Horizon 2020 ECSEL Joint Undertaking project SECREDAS, grant agreement n°783119.

References

1. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A subversion-resistant SNARK. In: ASIACRYPT (3). LNCS, vol. 10626, pp. 3–33. Springer (2017)
2. Alexandra Institute: FRESCO - a FRamework for Efficient Secure COmputation (2019), https://github.com/aicis/fresco
3. Aly, A., Keller, M., Rotaru, D., Scholl, P., Smart, N.P., Wood, T.: SCALE-MAMBA (2019), https://homes.esat.kuleuven.be/~nsmart/SCALE/

[4] https://www.unboundtech.com/usecase/virtual-hsm/

4. Baldimtsi, F., Camenisch, J., Dubovitskaya, M., Lysyanskaya, A., Reyzin, L., Samelin, K., Yakoubov, S.: Accumulators with applications to anonymity-preserving revocation. In: EuroS&P. pp. 301–315. IEEE (2017)
5. Baldimtsi, F., Canetti, R., Yakoubov, S.: Universally composable accumulators. IACR Cryptology ePrint Archive **2018**, 1241 (2018)
6. Baric, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: EUROCRYPT. LNCS, vol. 1233, pp. 480–494. Springer (1997)
7. Bauer, A., Herbst, N., Spinner, S., Ali-Eldin, A., Kounev, S.: Chameleon: A hybrid, proactive auto-scaling mechanism on a level-playing field. IEEE Trans. Parallel Distrib. Syst. **30**(4), 800–813 (2019)
8. Bellare, M., Fuchsbauer, G., Scafuro, A.: Nizks with an untrusted CRS: security in the face of parameter subversion. In: ASIACRYPT (2). LNCS, vol. 10032, pp. 777–804 (2016)
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: CRYPTO (3). LNCS, vol. 11694, pp. 701–732. Springer (2019)
10. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: EUROCRYPT. LNCS, vol. 765, pp. 274–285. Springer (1993)
11. Bowe, S., Gabizon, A., Miers, I.: Scalable multi-party computation for zk-snark parameters in the random beacon model. IACR Cryptology ePrint Archive **2017**, 1050 (2017)
12. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: CRYPTO. LNCS, vol. 2442, pp. 61–76. Springer (2002)
13. Campanelli, M., Gennaro, R., Goldfeder, S., Nizzardo, L.: Zero-knowledge contingent payments revisited: Attacks and payments for services. In: ACM CCS. pp. 229–243. ACM (2017)
14. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. pp. 136–145. IEEE (2001)
15. Cramer, R., Damgård, I., Escudero, D., Scholl, P., Xing, C.: SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In: CRYPTO (2). LNCS, vol. 10992, pp. 769–798. Springer (2018)
16. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In: ESORICS. LNCS, vol. 8134, pp. 1–18. Springer (2013)
17. Damgård, I., Koprowski, M.: Practical threshold RSA signatures without a trusted dealer. In: EUROCRYPT. LNCS, vol. 2045, pp. 152–165. Springer (2001)
18. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO. LNCS, vol. 7417, pp. 643–662. Springer (2012)
19. Derler, D., Hanser, C., Slamanig, D.: Revisiting cryptographic accumulators, additional properties and relations to other primitives. In: CT-RSA. LNCS, vol. 9048, pp. 127–144. Springer (2015)
20. Derler, D., Pöhls, H.C., Samelin, K., Slamanig, D.: A general framework for redactable signatures and new constructions. In: ICISC. LNCS, vol. 9558, pp. 3–19. Springer (2015)
21. Derler, D., Ramacher, S., Slamanig, D.: Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In: PQCrypto. LNCS, vol. 10786, pp. 419–440. Springer (2018)

22. Dowling, B., Günther, F., Herath, U., Stebila, D.: Secure logging schemes and certificate transparency. In: ESORICS (2). LNCS, vol. 9879, pp. 140–158. Springer (2016)
23. Frederiksen, T.K., Lindell, Y., Osheter, V., Pinkas, B.: Fast distributed RSA key generation for semi-honest and malicious adversaries. In: CRYPTO (2). LNCS, vol. 10992, pp. 331–361. Springer (2018)
24. Fuchsbauer, G.: Subversion-zero-knowledge snarks. In: PKC (1). LNCS, vol. 10769, pp. 315–347. Springer (2018)
25. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: ACM CCS. pp. 1179–1194. ACM (2018)
26. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In: ACNS. LNCS, vol. 9696, pp. 156–174. Springer (2016)
27. Kales, D., Omolola, O., Ramacher, S.: Revisiting user privacy for certificate transparency. In: EuroS&P. pp. 432–447. IEEE (2019)
28. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: ACM CCS. pp. 525–537. ACM (2018)
29. Keller, M., Orsini, E., Scholl, P.: Actively secure OT extension with optimal overhead. In: CRYPTO (1). LNCS, vol. 9215, pp. 724–741. Springer (2015)
30. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: ACM CCS. pp. 830–842. ACM (2016)
31. Laurie, B.: Certificate transparency. ACM Queue **12**(8), 10–19 (2014)
32. Laurie, B., Langley, A., Käsper, E.: Certificate transparency. RFC **6962**, 1–27 (2013)
33. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In: EUROCRYPT (2). LNCS, vol. 9666, pp. 1–31. Springer (2016)
34. Lindell, Y.: Fast secure two-party ECDSA signing. In: CRYPTO (2). LNCS, vol. 10402, pp. 613–644. Springer (2017)
35. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: ACM CCS. pp. 1837–1854. ACM (2018)
36. Lipmaa, H.: Secure accumulators from euclidean rings without trusted setup. In: ACNS. LNCS, vol. 7341, pp. 224–240. Springer (2012)
37. Merkle, R.C.: A certified digital signature. In: CRYPTO. LNCS, vol. 435, pp. 218–238. Springer (1989)
38. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed e-cash from bitcoin. In: IEEE S&P. pp. 397–411. IEEE (2013)
39. Mohassel, P., Rindal, P.: Aby^3: A mixed protocol framework for machine learning. In: ACM CCS. pp. 35–52. ACM (2018)
40. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: IEEE S&P. pp. 19–38. IEEE (2017)
41. Nguyen, L.: Accumulators from bilinear pairings and applications. In: CT-RSA. LNCS, vol. 3376, pp. 275–292. Springer (2005)
42. Orsini, E., Smart, N.P., Vercauteren, F.: Overdrive2k: Efficient secure MPC over Z2k from somewhat homomorphic encryption. IACR Cryptology ePrint Archive **2019**, 153 (2019)
43. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Authenticated hash tables. In: ACM CCS. pp. 437–448. ACM (2008)

44. Pöhls, H.C., Samelin, K.: On updatable redactable signatures. In: ACNS. LNCS, vol. 8479, pp. 457–475. Springer (2014)
45. Sander, T.: Efficient accumulators without trapdoor extended abstracts. In: ICICS. LNCS, vol. 1726, pp. 252–262. Springer (1999)
46. Smart, N.P., Alaoui, Y.T.: Distributing any elliptic curve based protocol: With an application to mixnets. In: IMACC (2019), (to appear)
47. University of Bristol: Multi-Protocol SPDZ (2019), https://github.com/data61/MP-SPDZ
48. Wagh, S., Gupta, D., Chandran, N.: Securenn: 3-party secure computation for neural network training. PoPETs **2019**(3), 26–49 (2019)